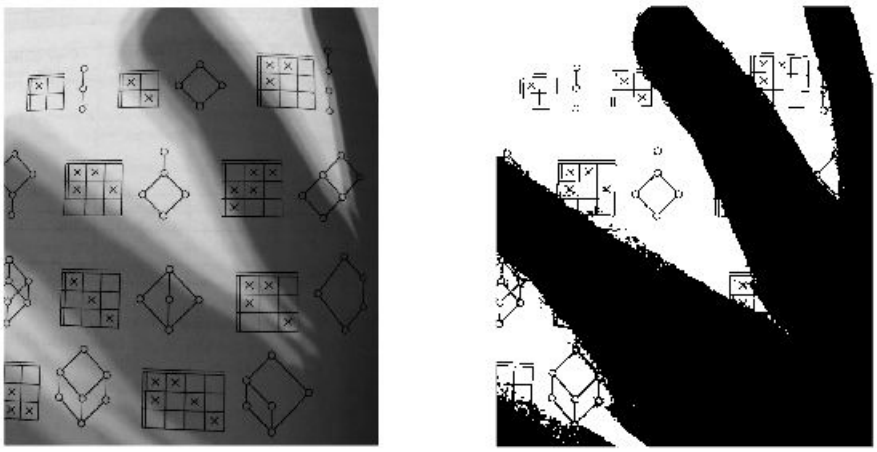
Cvičení 4 - Transformace intenzit II

Prahování

Lokální prahování

Pro některé obrázky není možné najít optimální prahovou hodnotu.

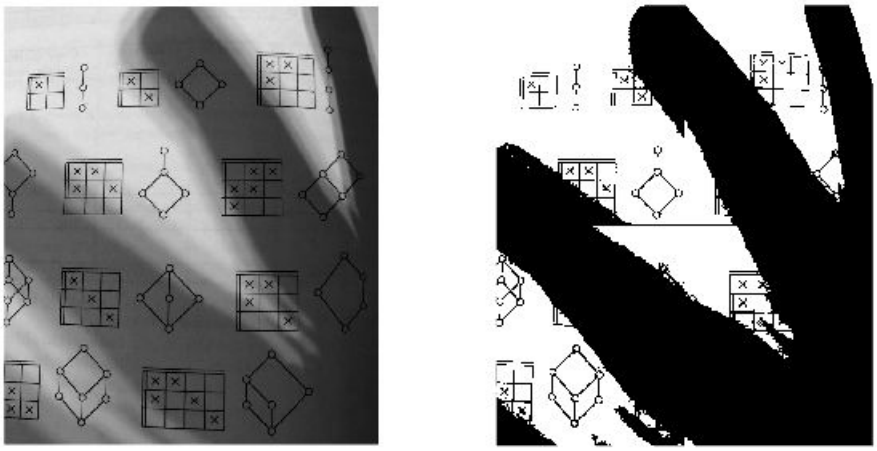
```
f3 = imread('lokalni.jpg');
g3 = imbinarize(f3);

figure,
subplot(1,2,1), imshow(f3);
subplot(1,2,2), imshow(g3);
```



Obraz můžeme rozdělit na více částí a hledat optimální prahovou hodnotu pro ně.

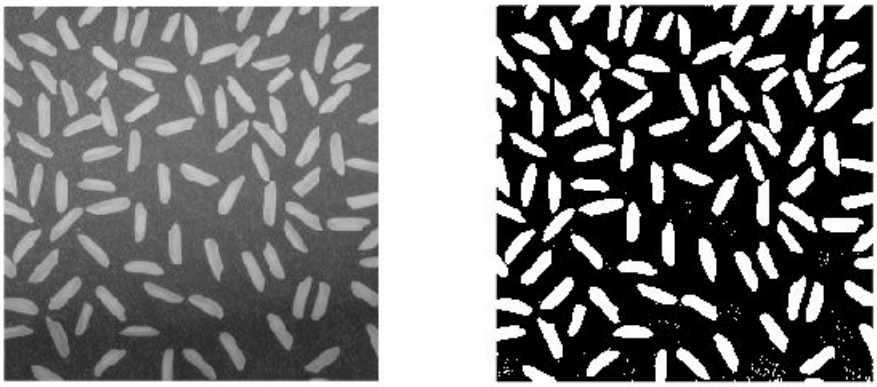
```
f4 = f3;
[m,n]=size(f3);
f4(1:floor(m/2), 1:floor(n/2)) = imbinarize(f4(1:floor(m/2), 1:floor(n/2)));
f4(1:floor(m/2), floor(n/2) : end) = imbinarize(f4(1:floor(m/2), floor(n/2) : end));
f4(floor(m/2) : end, 1:floor(n/2)) = imbinarize(f4(floor(m/2) : end, 1:floor(n/2)));
f4(floor(m/2) : end, floor(n/2) : end) = imbinarize(f4(floor(m/2) : end,
floor(n/2) : end));

figure,
```

```
subplot(1,2,1), imshow(f3);

subplot(1,2,2), imshow(f4,[]);
```



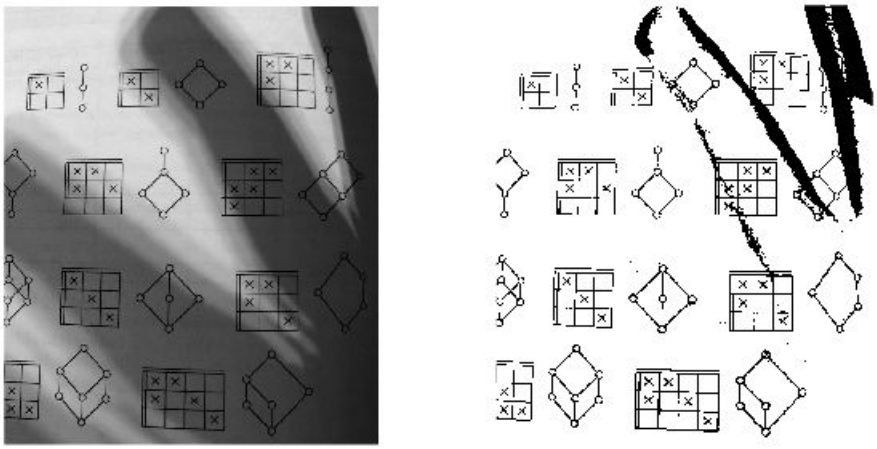
Lepší výsledky ale dává takzvané adaptivní prahování. Ideální prahová hodnota se hledá pro každý pixel zvlášť. Pro každý pixel vezmeme nějaké jeho okolí a pro něj spočítáme ideální prahovou hodnotu, například pomocí metody Otsu. V matlabu můžeme pro adaptivní prahování použít `imbinarize()` následujícím způsobem.

```
f3 = imread('rice.png');
g3 = imbinarize(f3,'adaptive');

figure,
subplot(1,2,1), imshow(f3);
subplot(1,2,2), imshow(g3);
```



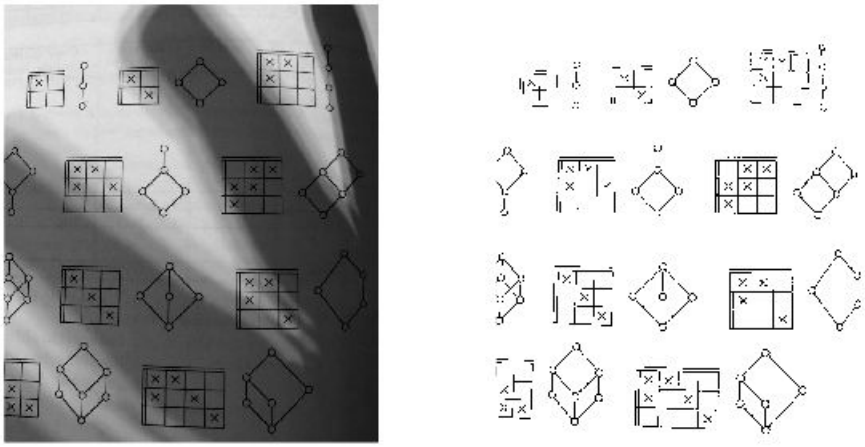
Pokud je popředí obrázku tmavší, je potřeba nastavit další parametry. Například následovně.

```
f3 = imread('lokalni.jpg');
g3 = imbinarize(f3,'adaptive','ForegroundPolarity','dark','Sensitivity',0.4);

figure,
subplot(1,2,1), imshow(f3);
subplot(1,2,2), imshow(g3);
```



Jednou z nejpoužívanějších metod pro lokální (adaptivní) prahování je metoda založená na standardní odchylce počítané pro každý bod v obraze. Tato hodnota se počítá oproti bodům v okolí (nejčastěji velikosti 3x3). Tento způsob má však více společného s filtrováním (operacemi s okolím bodu), kterým se budeme věnovat později. Zde bez dalšíhovysvětlení jen přikládám kód.

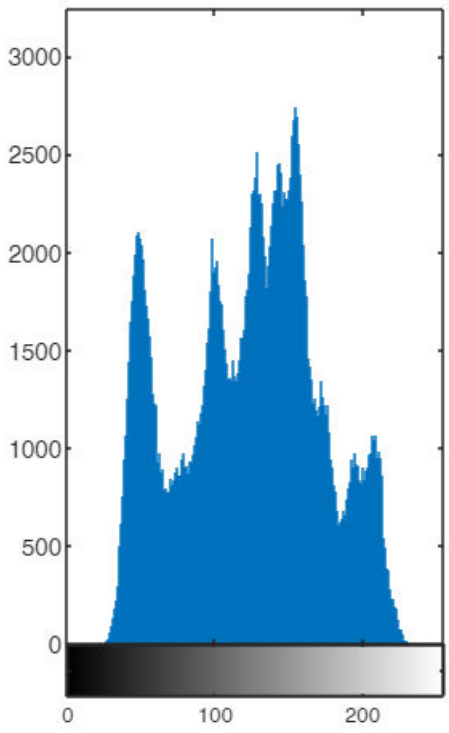
```
f3 = imread('lokalni.jpg');
f4=im2double(f3);
nhood = ones(3)/9;
g_std = stdfilt(f4);
g_mean = imfilter(f4,nhood);

a=0.3;
b=1-a;
T = a*g_std + b*g_mean;

g4 = imbinarize(f4,T);

figure,
subplot(1,2,1), imshow(f4);
subplot(1,2,2), imshow(g4);
```



Histogram obrazku

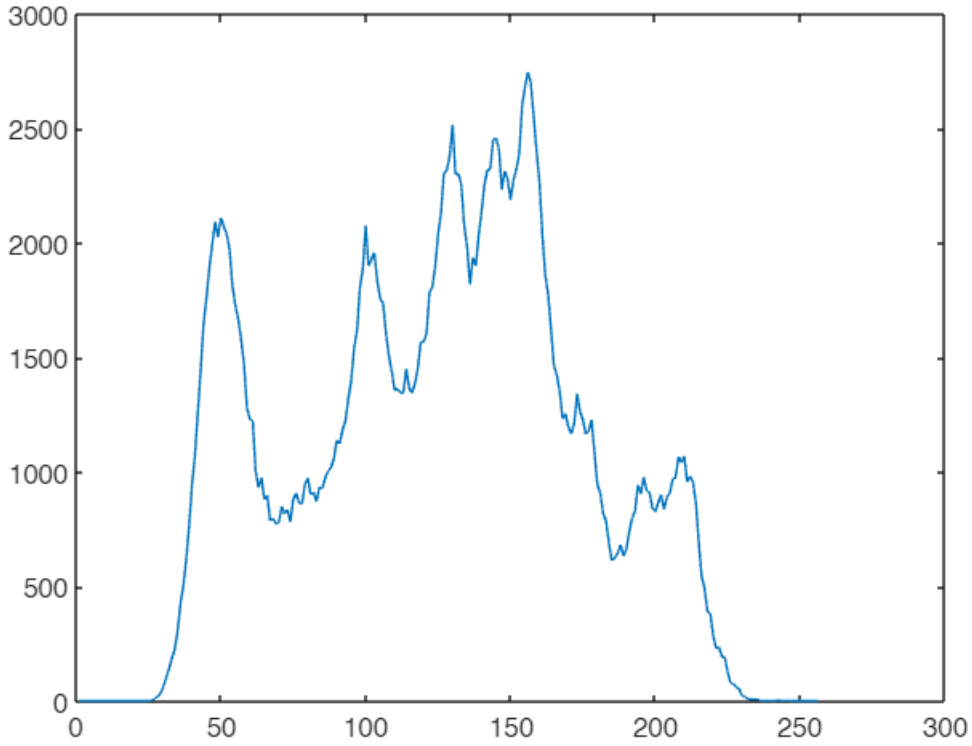
K popisu obrazu můžeme použít histogram. Je to diskrétní funkce, která každé intenzitě přiřadí počet pixelů v obraze s touto intenzitou. V matlabu můžeme histogram vykreslit pomocí funkcí `histogram() nebo imhist()`. Jak jsme si ukázali minule.

```
I = imread('lena_gray.png');
figure,
subplot(1,2,1), imshow(I);
subplot(1,2,2), imhist(I)
```



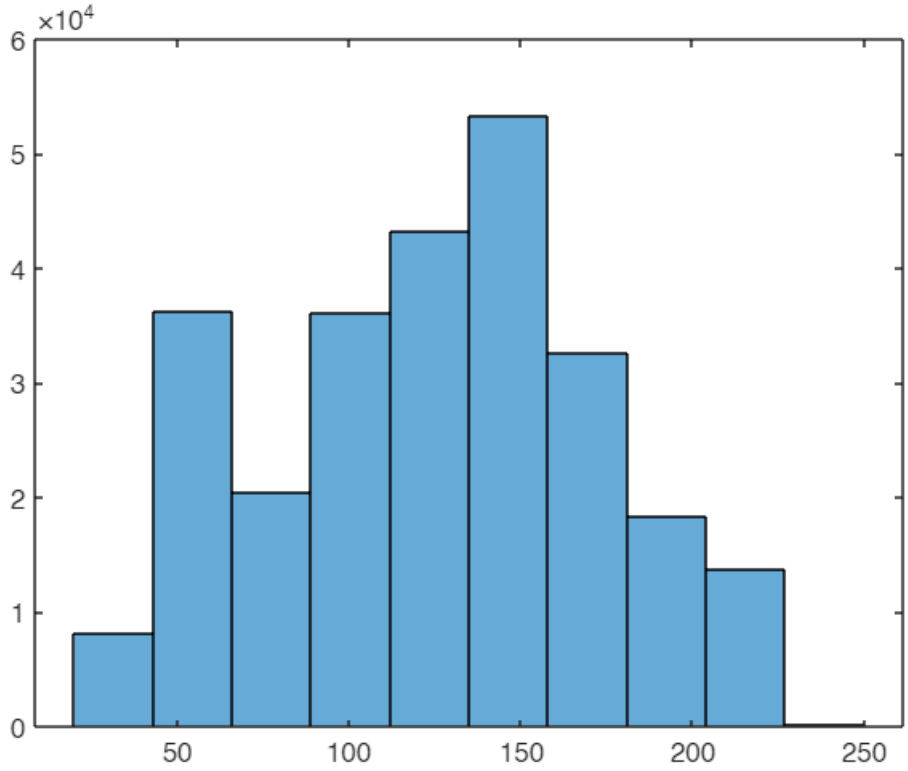
Případně si hodnoty uložit do nějaké proměnné a tu pak vykreslit.

```
histogramI = imhist(I);
figure, plot(histogramI);
```



Funkce histogram() se používá zejména pokud chceme takzvaný binding histogram (viz přednáška).

```
pocet_prihradek = 10;
figure, histogram(I,pocet_prihradek);
```



Také funkci imhist() je možné zadat počet přihrádek a tím získat binding histogram.

ÚKOL 1

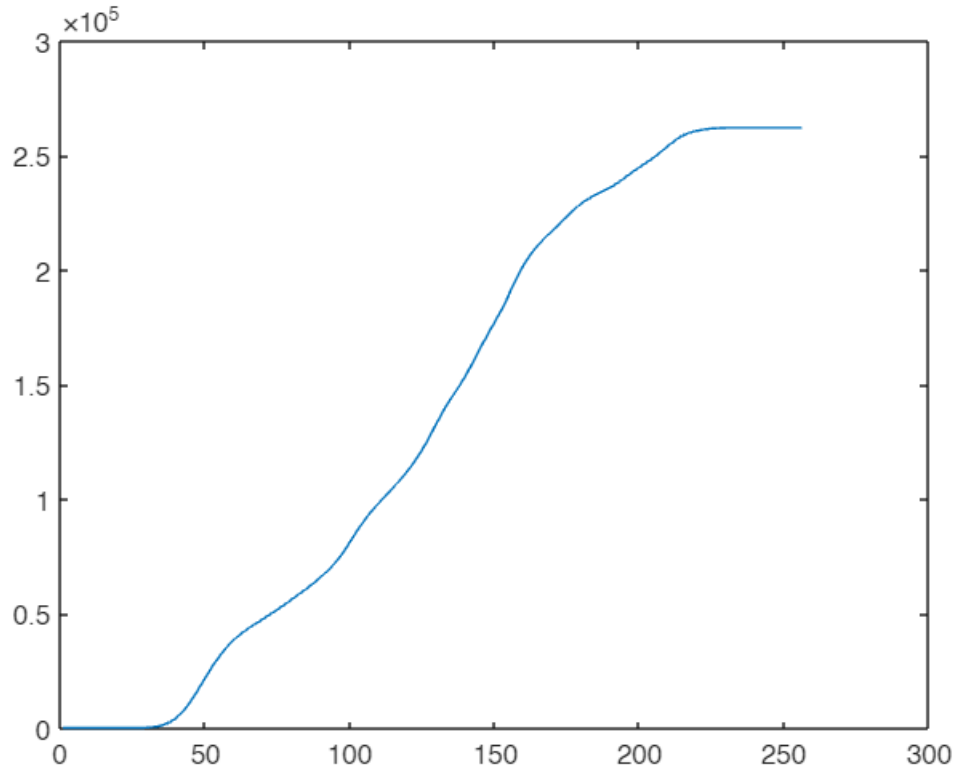
Jak bychom spočítali binding histogram z histogramu `histogramI`?

```
pocet_prihradek = 10;
histogramI = imhist(I);
```

Kumulativní histogram

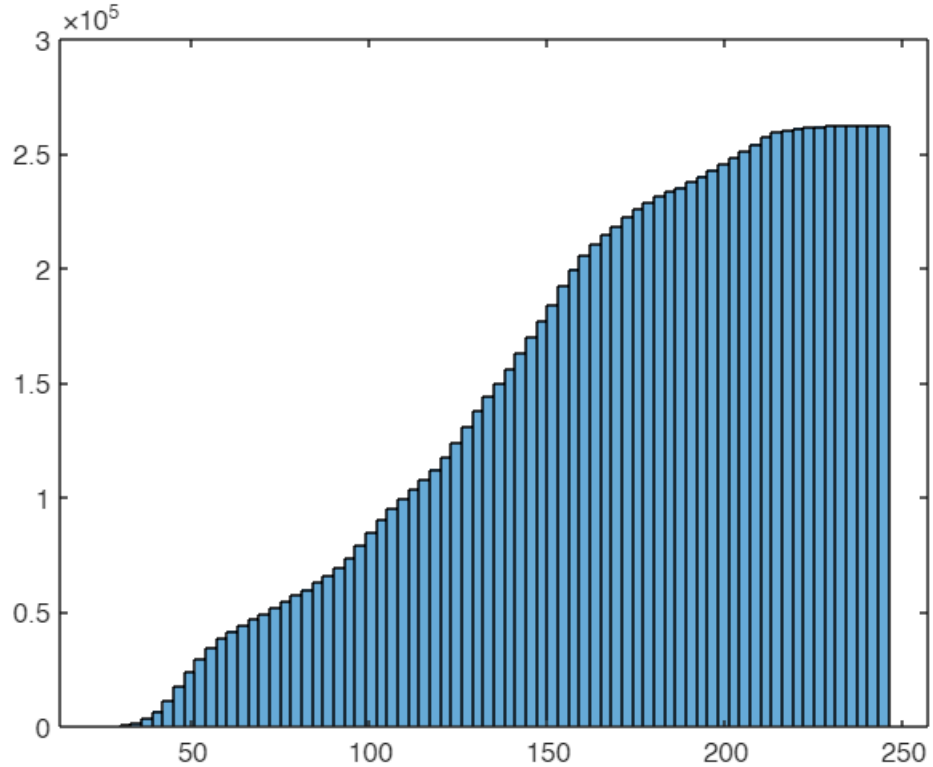
Kumulativní histogram uchovává informaci, kolik pixelů v obrázku má intenzitu menší rovnu každé intenzitě. Z histogramu vytvoříme kumulativní histogram za pomoci funkce `cumsum()`.

```
[pocet,~] = imhist(I);
cumh = cumsum(pocet);
figure, plot(cumh);
```



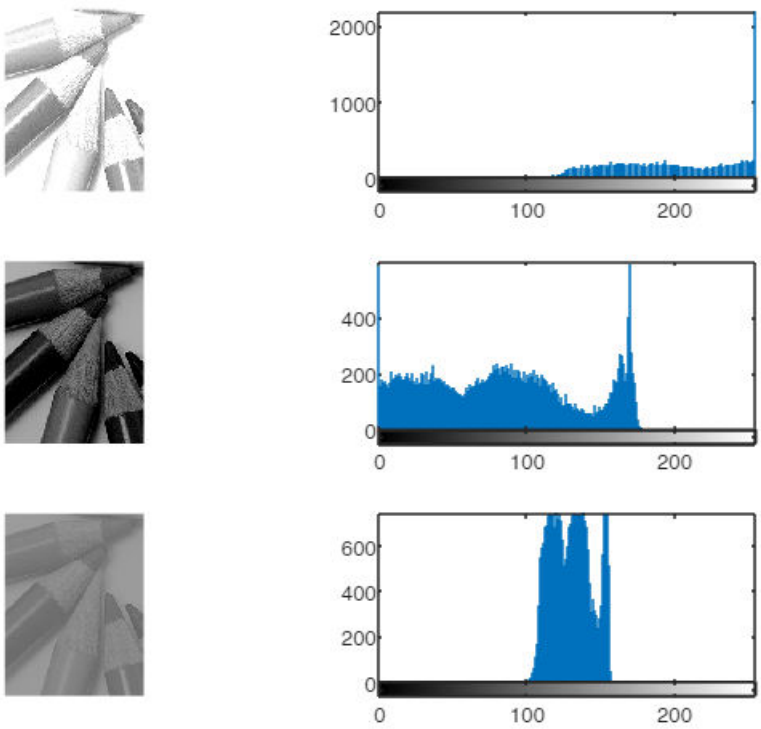
Připadně můžeme využít funkci histogram() následujícím způsobem.

```
figure, histogram(I,'Normalization','cumcount');
```



Histogram - co z něj vyčteme?

```
I1 = imread('p1.png');
I2 = imread('p2.png');
I3 = imread('p3.png');

figure,
subplot(3,2,1)
imshow(I1);
subplot(3,2,2)
imhist(I1)

subplot(3,2,3)
imshow(I2);
subplot(3,2,4)
imhist(I2)

subplot(3,2,5)
imshow(I3);
subplot(3,2,6)
imhist(I3)
```



Z histogramu se dá vyčíst několik vlastností obrázku, jako je například zda je příliš světlý, tmavý, nebo má nízký kontrast.

Úkol 2 z minulého cvičení

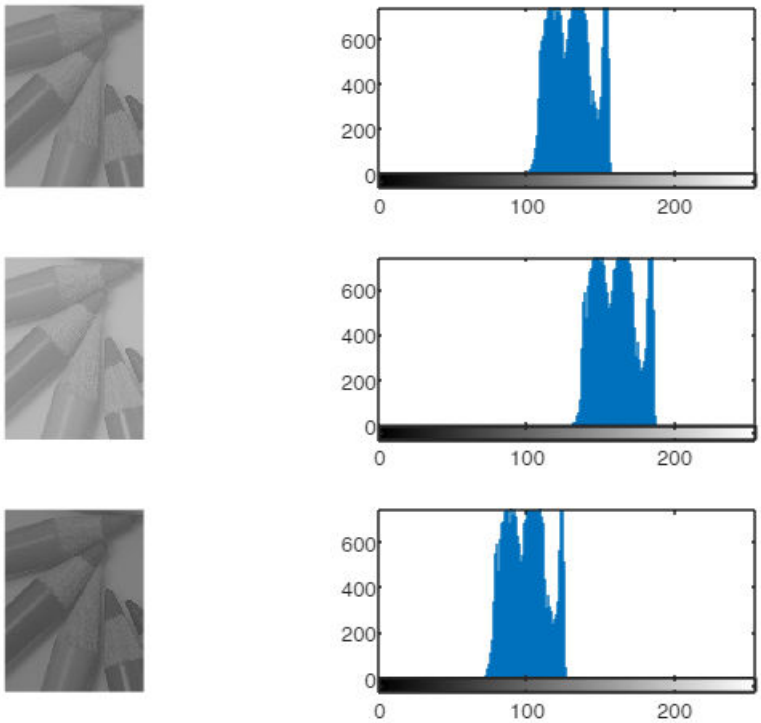
Jak ovlivní histogram obrázku jasové transformace?

Jas

Histogram se posune vlevo (snížení jasu) nebo vpravo (zvýšení jasu)

```
f = imread('p3.png');
k1=30;
k2 = -30;



g1 = f + k1;

g2 = f + k2;

figure,
subplot(3,2,1)
imshow(f);
subplot(3,2,2)
imhist(f)
```

```
subplot(3,2,3)
imshow(g1);
subplot(3,2,4)
imhist(g1)
subplot(3,2,5)
imshow(g2);
subplot(3,2,6)
imhist(g2)
```

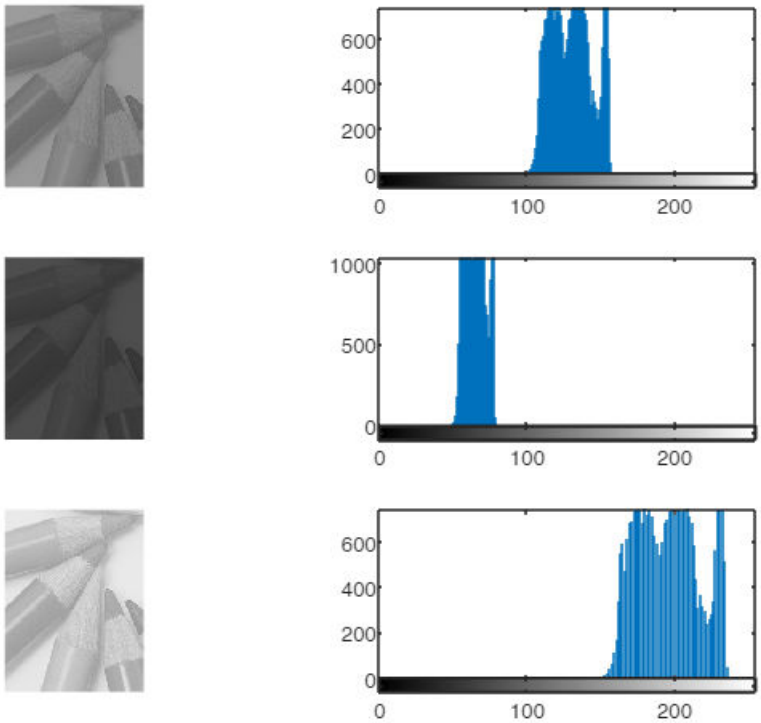


Kontrast

Histogram se zúží (snížení kontrastu) nebo roztáhne (zvýšení).

```
c1=0.5;
c2 = 1.5;

g3 = c1*f;
g4 =c2*f;

figure,
subplot(3,2,1)
imshow(f);
subplot(3,2,2)
imhist(f)
subplot(3,2,3)
imshow(g3);
```

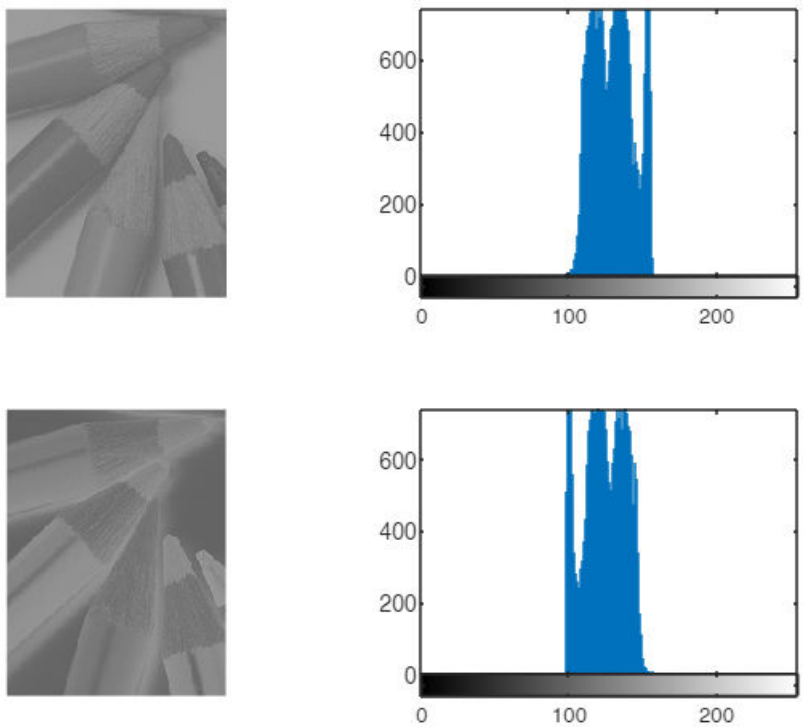
```
subplot(3,2,4)
imhist(g3)
subplot(3,2,5)
imshow(g4);
subplot(3,2,6)
imhist(g4)
```



Negativ obrázku

Histogram se převrátí.

```
g5 =255 - f;

figure,
subplot(2,2,1)
imshow(f);
subplot(2,2,2)
imhist(f)
subplot(2,2,3)
imshow(g5);
subplot(2,2,4)
imhist(g5)
```

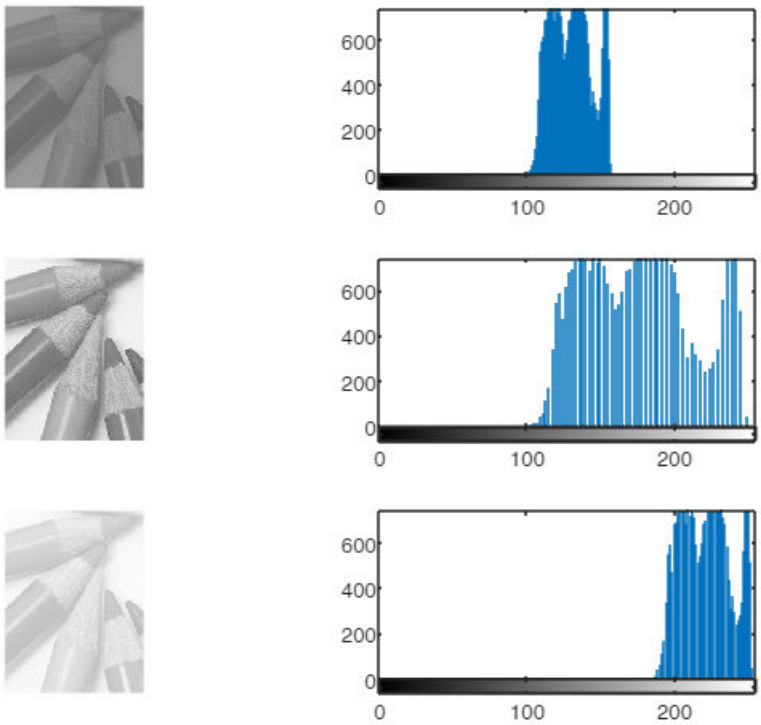


Gamma korekce

Levá (pravá) část histogramu se zůží a pravá (levá) roztáhne.

```
gam1 = 2;
gam2 = 0.7;

g6 = double(f).^gam1;
g7 = double(f).^gam2;

g6 = uint8(255*(g6/max(max(g6))));
g7 = uint8(255*(g7/max(max(g7))));

figure,
subplot(3,2,1)
imshow(f);
subplot(3,2,2)
imhist(f)
subplot(3,2,3)
imshow(g6);
subplot(3,2,4)
imhist(g6)
subplot(3,2,5)
imshow(g7);
subplot(3,2,6)
```

```
imhist(g7)
```



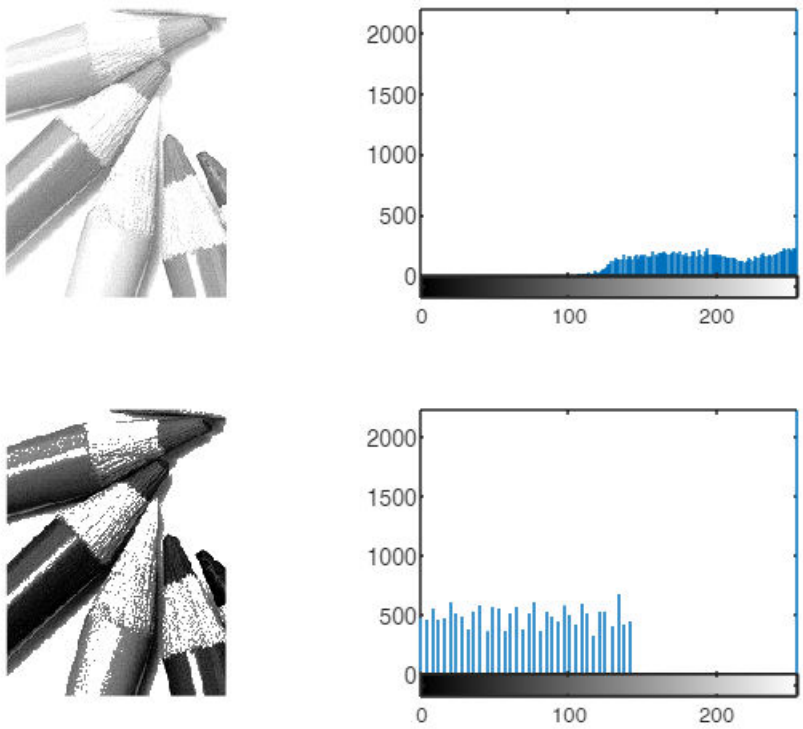
Transformace založené na histogramu

Na první pohled by se mohlo zdát, že ideální histogram obrázku vypadá tak, že je rozložen pravidelně mezi všechny intenzity (všechny intenzity jsou v obraze zastoupeny stejně často).

Na této myšlence je založena myšlenka operace vyváženého histogramu. V matlabu k této operaci slouží funkce `histeq()`.

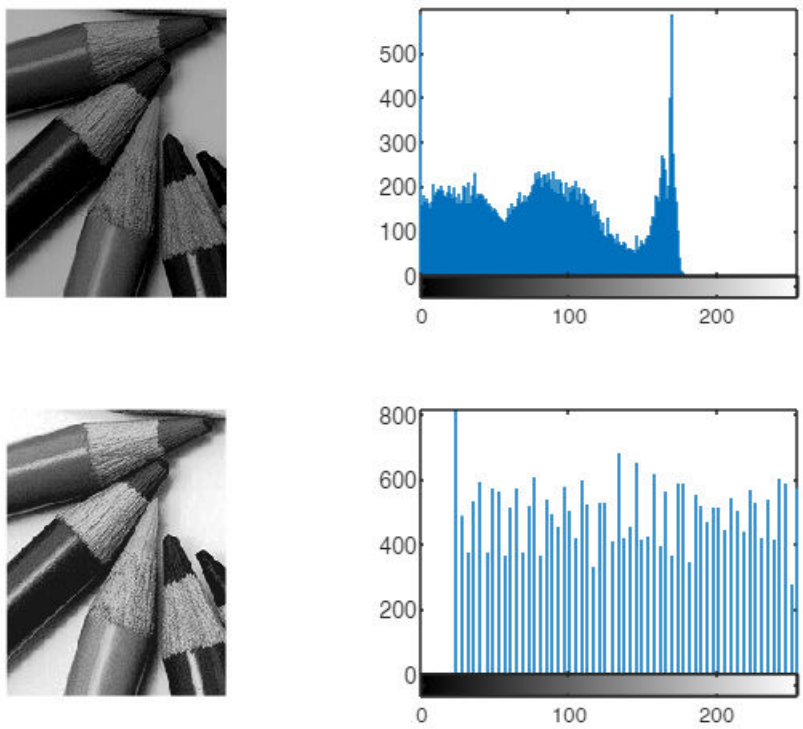
Obrazek 1

```
J1 = histeq(I1);
J2 = histeq(I2);
J3 = histeq(I3);

figure,
subplot(2,2,1)
imshow(I1);
subplot(2,2,2)
imhist(I1)
subplot(2,2,3)
imshow(J1);
subplot(2,2,4)
imhist(J1)
```



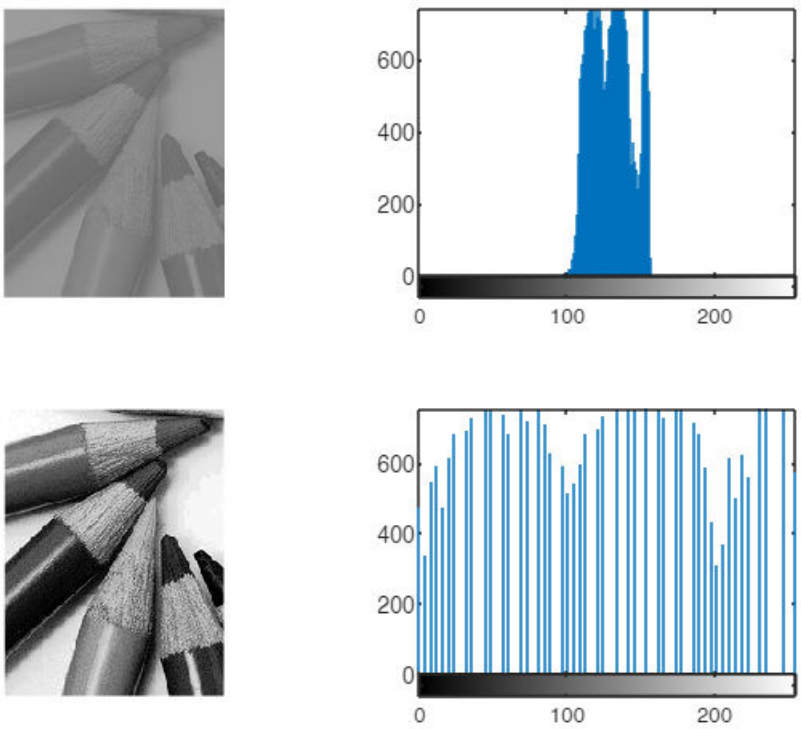
Obrázek 2

```
figure,
subplot(2,2,1)
imshow(I2);
subplot(2,2,2)
imhist(I2)
subplot(2,2,3)
imshow(J2);
subplot(2,2,4)
imhist(J2)
```



Obrázek 3

```
figure,
subplot(2,2,1)
imshow(I3);
subplot(2,2,2)
imhist(I3)
subplot(2,2,3)
imshow(J3);
subplot(2,2,4)
imhist(J3)
```



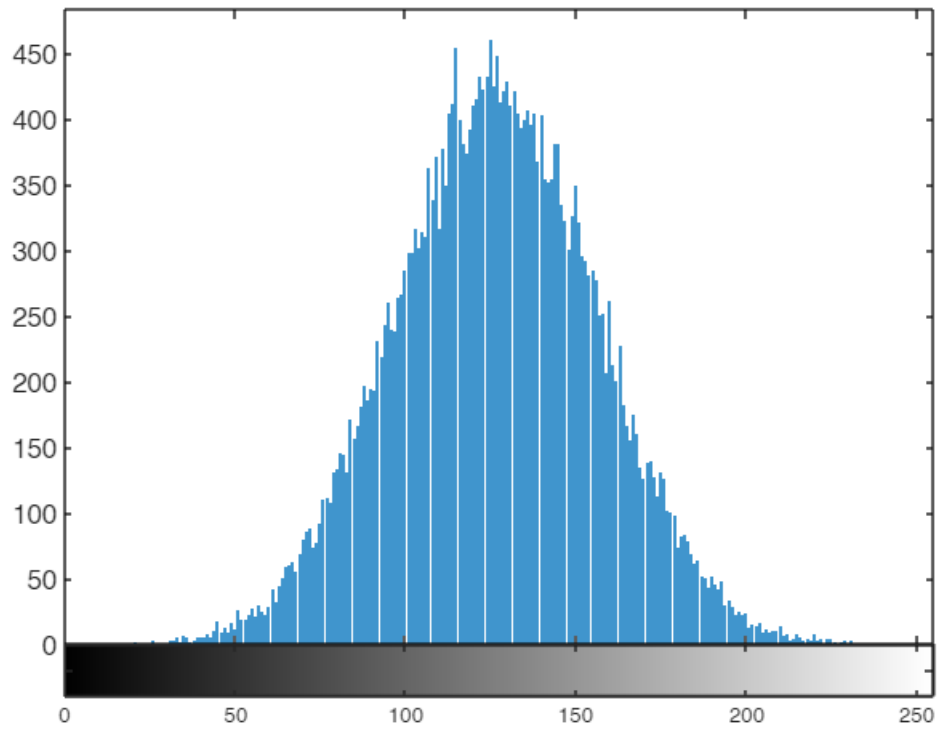
Specifikace histogramu

Obecně to však nemusí platit pro všechny obrázky, například pro obrázek se šachovnicí.

Obdobným způsobem, jak pracuje metoda vyvážení histogramu pracuje i operace specifikace histogramu. Zde se nesnažíme, aby výsledný obrázek měl vyvážený histogram, ale aby se co nejvíce podobal námi specifikovanému hostogramu.

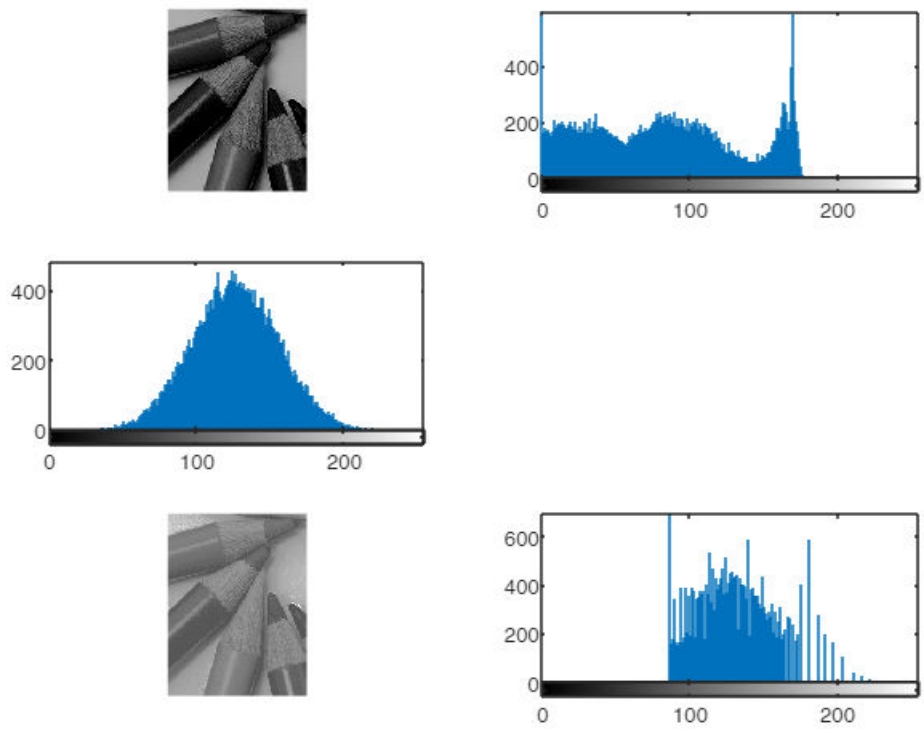
V následujícím příkladu chceme, aby výsledný tvat histogramu byl gausovská křivka (tu záskáme pomocí funkce rand, která vrací náhodně hodnoty s gausovským rozložením).

```
spechistobr = randn(size(I2));
minimum = min(min(spechistobr));
spechistobr = spechistobr + (0-minimum);
maximum = max(max(spechistobr));
spechistobr = 255*(spechistobr/maximum);
spechistobr = uint8(round(spechistobr));

figure,
imhist(spechistobr)
```



Pro operaci specifikace histogramu opět využijeme funkci `histeq()`, které jako druhý argument předáme histogram, ke kterému se chceme přiblížit.

```
[COUNTS,X] = imhist(spechistobr);
Jspec = histeq(I2, COUNTS);

figure,
subplot(3,2,1)
imshow(I2);
subplot(3,2,2)
imhist(I2)
subplot(3,2,3)
imhist(spechistobr)
subplot(3,2,5)
imshow(Jspec);
subplot(3,2,6)
imhist(Jspec)
```



Jasové transformace barevných obrázků

RGB barevný model

```
I = imread('pastelky.png');
size(I)
```

```
ans = 1×3
   426   640     3
```

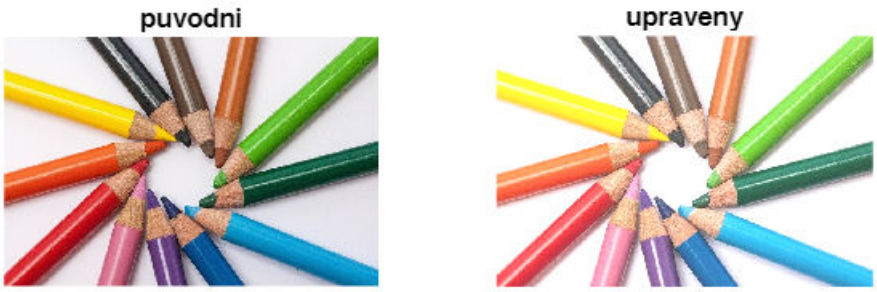
```
% jednotlive slozky
I_R = I(:, :, 1); % cervena
I_G = I(:, :, 2); % zelena
I_B = I(:, :, 3); % modra

figure,
imshow(I);
```

```
figure,
subplot(1,3,1), imshow(I_R);
title('cervena slozka');
subplot(1,3,2), imshow(I_G);
title('zelena slozka');
subplot(1,3,3), imshow(I_B);
title('modra slozka');
```

Jasové operace aplikujeme na všechny složky (na každou zvlášť).

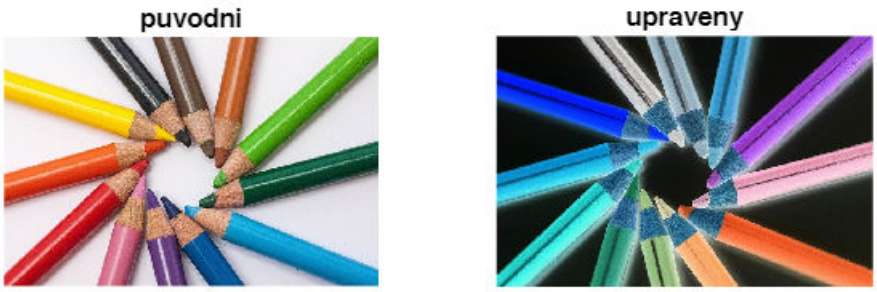
Změna jasu

```
J = I;
k = 50;
J(:,:,1) = J(:,:,1) + k;
J(:,:,2) = J(:,:,2) + k;
J(:,:,3) = J(:,:,3) + k;

figure,
subplot(1,2,1), imshow(I);
title('puvodni');
subplot(1,2,2), imshow(J);
title('upraveny');
```



Barevný komplement

```
J = I;

J(:,:,1) = 255-J(:,:,1);
J(:,:,2) = 255-J(:,:,2);
J(:,:,3) = 255-J(:,:,3);

figure,
subplot(1,2,1), imshow(I);
title('puvodni');
subplot(1,2,2), imshow(J);
title('upraveny');
```



Na každou složku můžeme aplikovat jinou transformaci. Například v případě transformace, která se nazývá barevná korekce. V ní aplikujeme gamma transformaci jen na jednu ze složek a zbylé dvě zůstanou beze změny.

ÚKOL 2

Naprogramujte barevnou korekci. Aplikujte jí na jednotlivé složky a výsledky porovnejte. Vyzkoušejte výsledky i pro různá gamma.

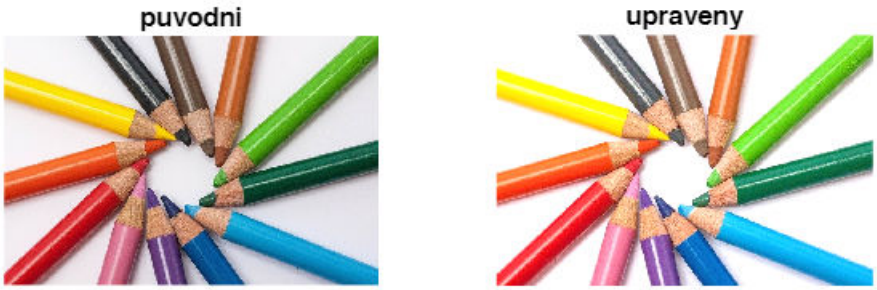
HSV barevný model

Zde upravujeme pouze jasovou složku V.

Pro převod mezi RGB a HSV modely použijeme funkce `rgb2hsv()` a `hsv2rgb()`.

```
I_hsv = rgb2hsv(I);

% Zmena jasu
I_hsv(:,:,3) = I_hsv(:,:,3) + k/255;

K = hsv2rgb(I_hsv);
figure,
subplot(1,2,1), imshow(I);
title('puvodni');
subplot(1,2,2), imshow(K);
```

```
title('upraveny');
```



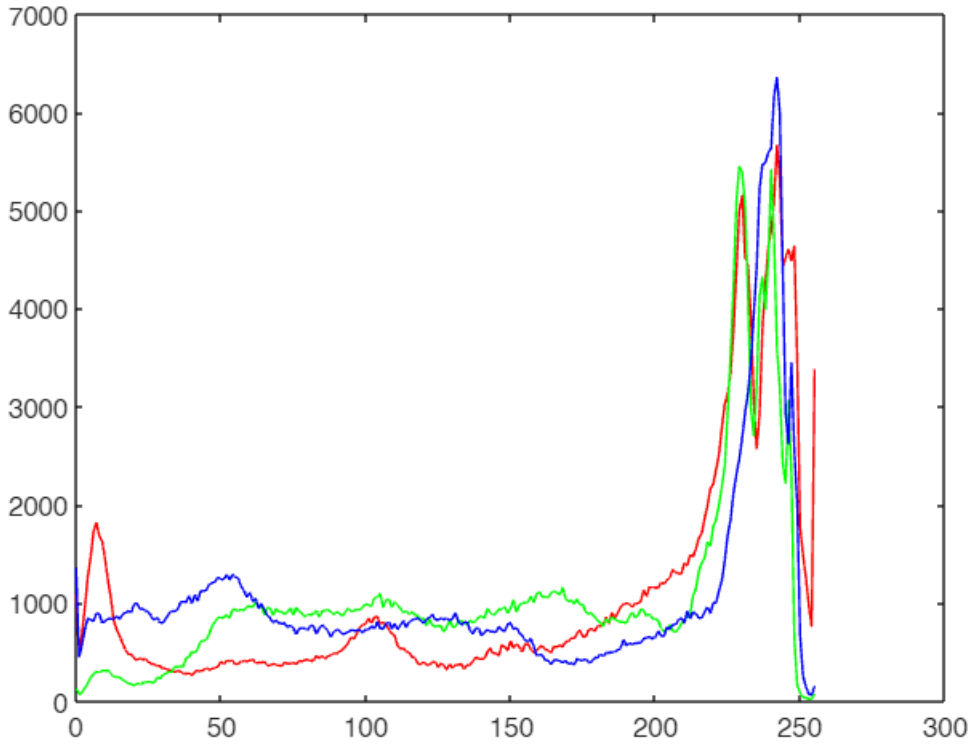
ÚKOL 3

Aplikujte změnu jasu na obrázek v RGB modelu i v modelu HSV a porovnejte výsledky.

Histogram barevných obrázků

histogram každé z barevných složek

```
hist_r = imhist(I(:,:,1));
hist_g = imhist(I(:,:,2));
hist_b = imhist(I(:,:,3));

figure,
plot(0:255, hist_r,"r-",0:255, hist_g,"g-",0:255,hist_b,"b-");
```



Vyvážení histogramu barevných obrázků

V RGB aplikujeme (`histeq()`) na každou barevnou složku zvlášť, v HSV na složku V.

ÚKOL 4

Aplikujte vyvážení histogramu na obrázek v RGB modelu i v modelu HSV a porovnejte výsledky.

V RGB nedostaneme hezký výsledek, objevují se odstíny, které v původním obrázku nejsou.