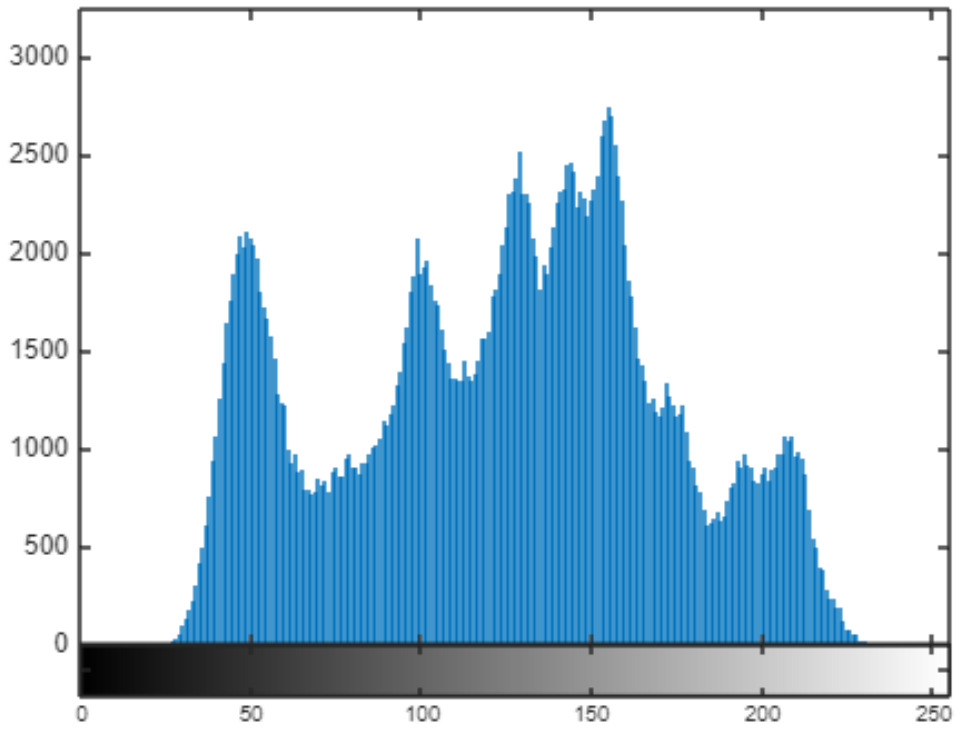
Cvičení 3 - Transformace intenzit

Histogram obrazku

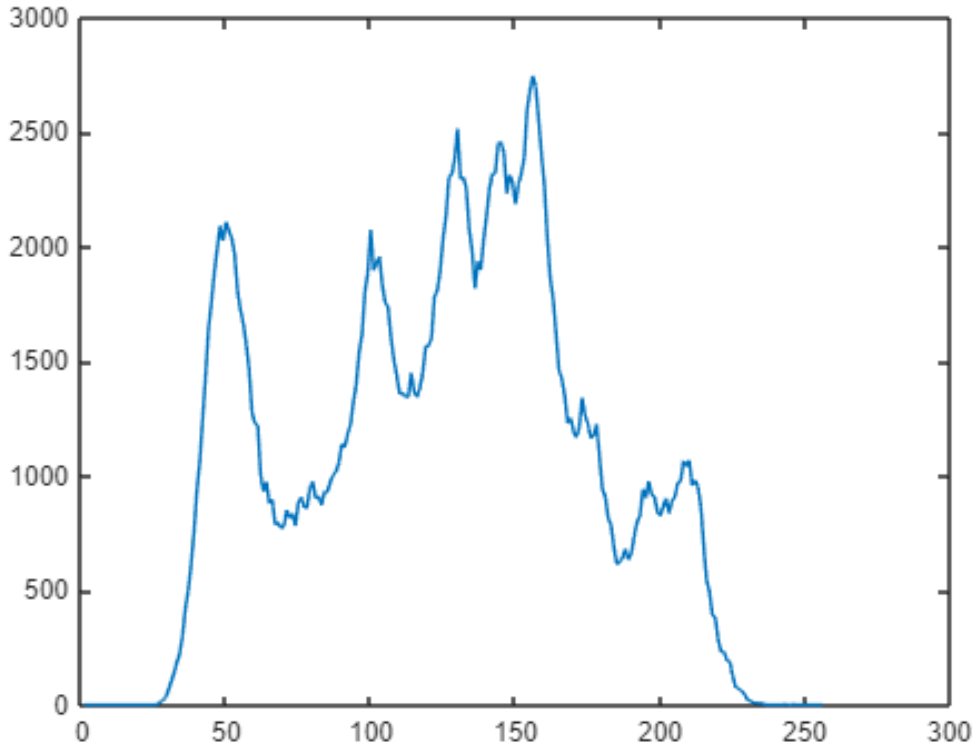
K popisu obrazu můžeme použít histogram. Je to diskrétní funkce, která každé intenzitě přiřadí počet pixelů v obraze s touto intenzitou. V matlabu můžeme histogram vykreslit pomocí funkcí `histogram() nebo imhist()`.

```
I = imread('lena_gray.png');
imhist(I)
```

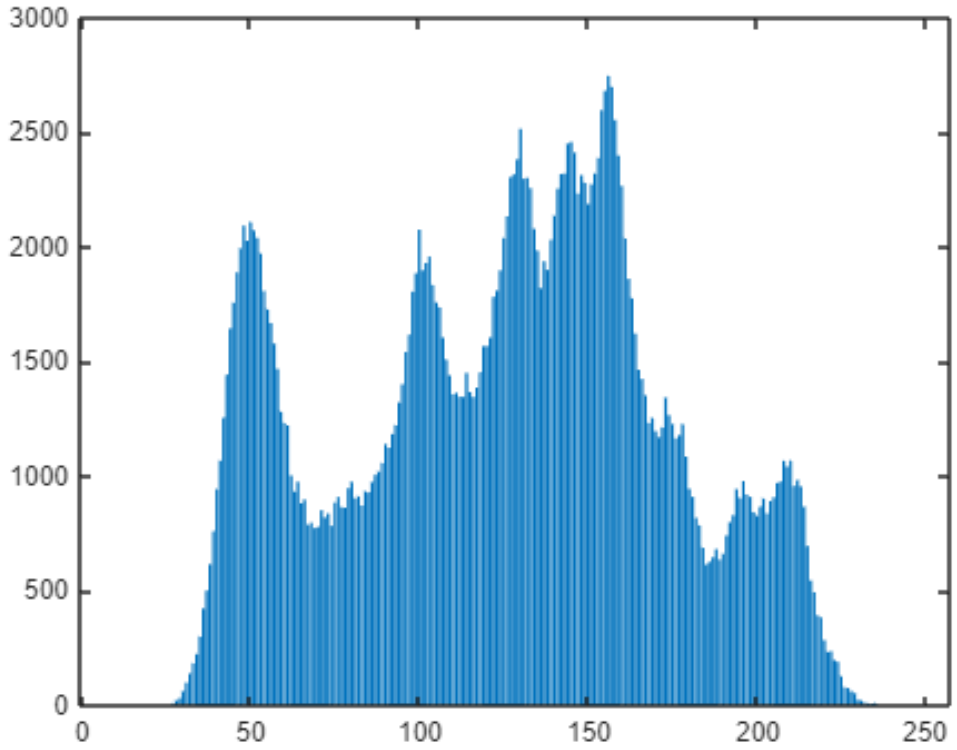


Případně si hodnoty uložit do nějaké proměnné a tu pak vykreslit.

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




```
figure, bar(histogramI);
```



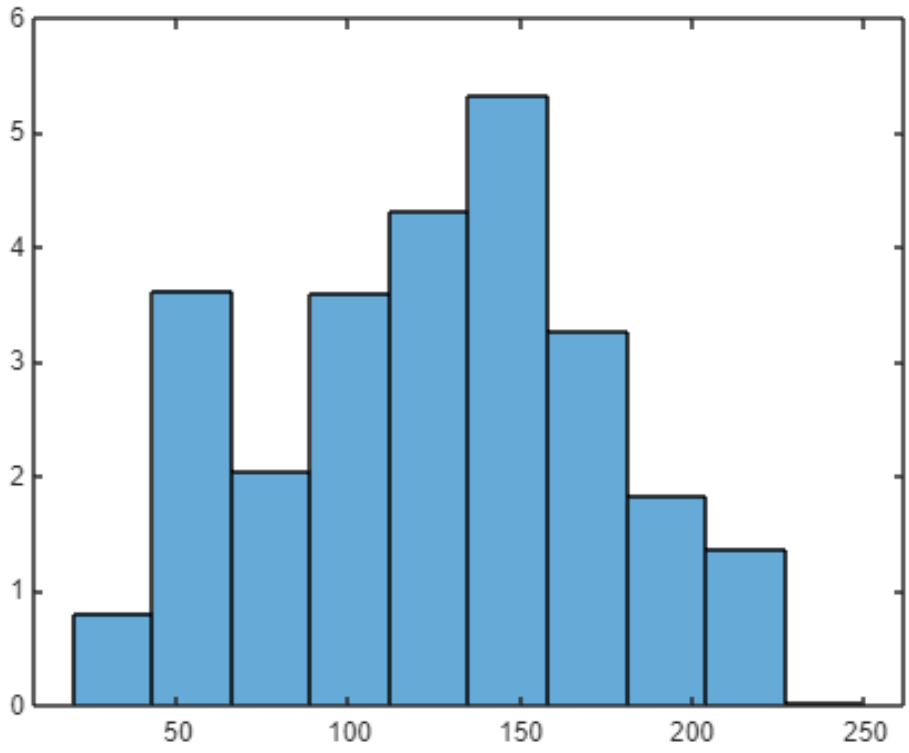
ÚKOL 1

Jak bychom spočítali binding histogram (viz přednáška) *z histogramu* `histogramI`?

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

Funkce histogram() se používá zejména pokud chceme binding histogram.

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

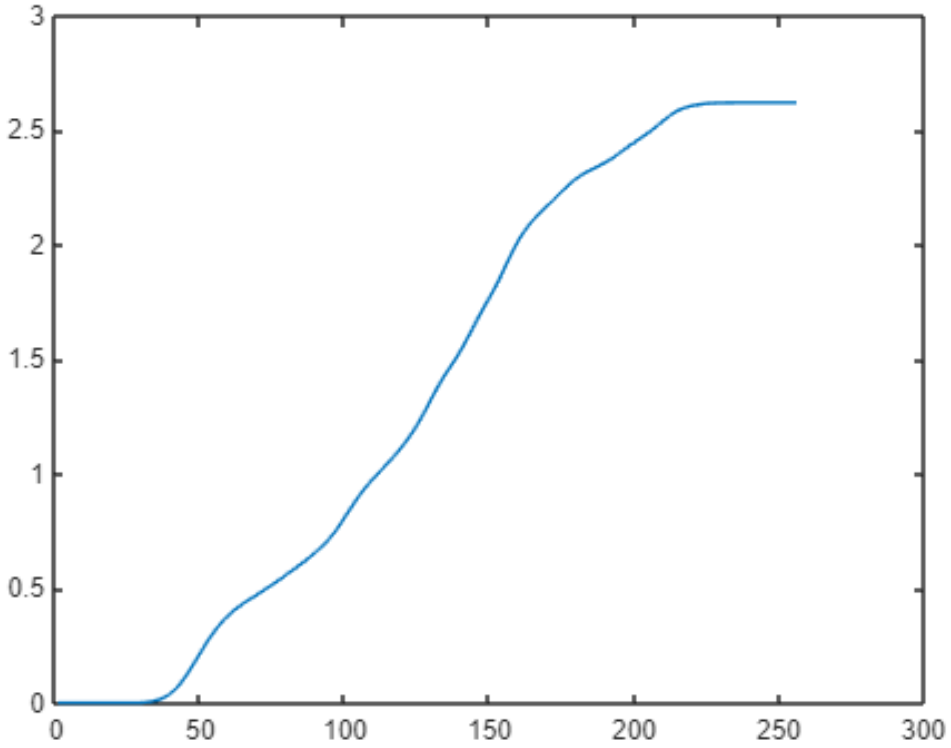


Také funkci imhist() je možné zadat počet přihrádek a tím získat binding histogram.

Kumulativní histogram

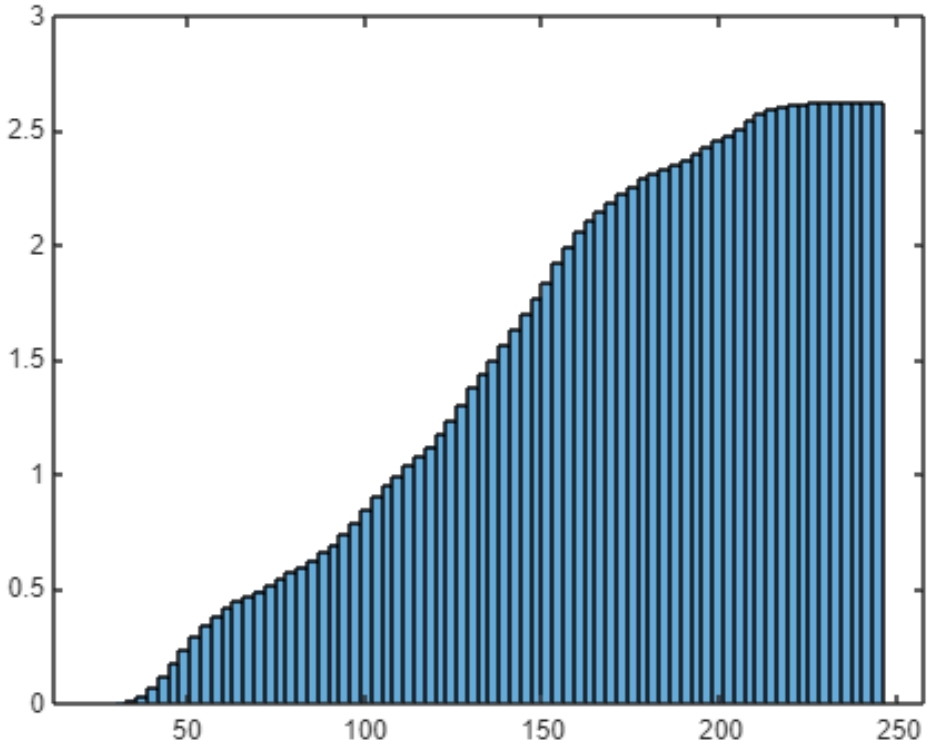
Kumulativní histogram uchovává informaci, kolik pixelů v obrázku má intenzitu menší rovnu každé intenzitě. Z histogramu vytvoříme kumulativní histogram za pomoci funkce `cumsum()`.

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



Připadně můžeme využít funkci histogram() následujícím způsobem.

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



Transformace intenzit

$g(x, y) = T[f(x, y)]$

f ... vstupní obrázek

g ... výstupní obrázek

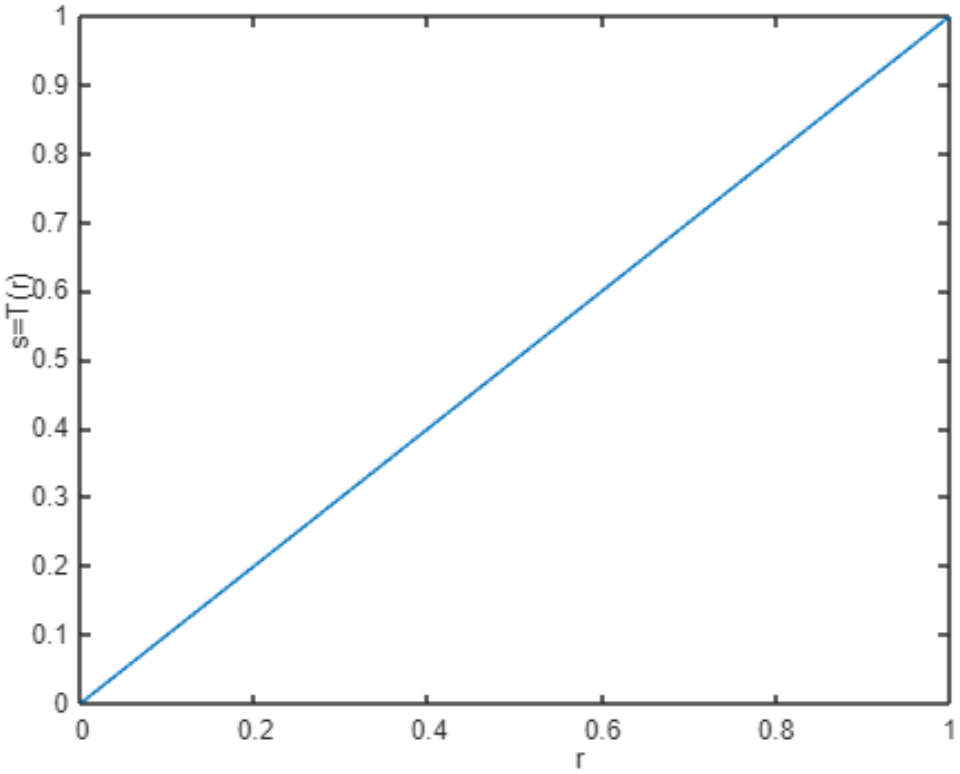
T - transformace barev

Transformaci zadáváme jako funkci (**transformační funkce**), která každé jasové hodnotě *r* přiřadí novou jasovou hodnotu *s*.

$s = T(r)$

Definice a vykreslení trasnformační funkce

```
r = (0:255)/255;
figure, plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
```



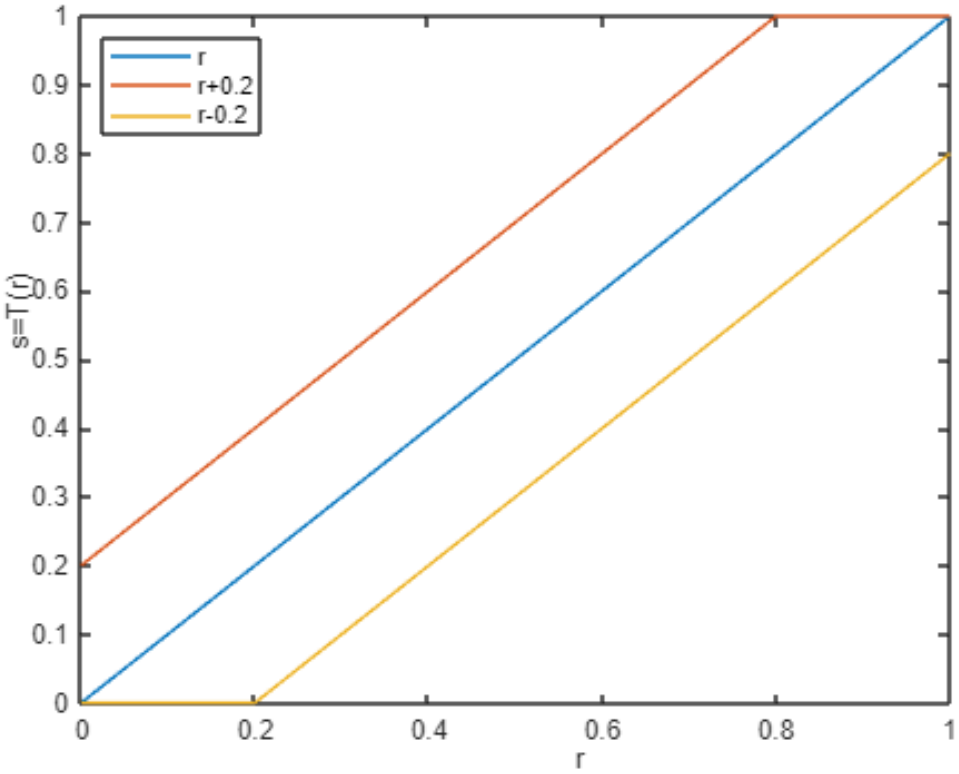
Pro obrázky typu uint8 jsou jasové hodnoty od 0 do 255. My tyto hodnoty převádíme na rozsah od 0 do 1.

Změna jasu

Transformační funkce je ve tvaru $s = r + k$, kde *k* je konstanta určující zda jas snižujeme, nebo zvyšujeme a jak moc. Výsledné hodnoty, které jsou menší než 0 jsou nastaveny na 0 a hodnoty vyšší než maximální intenzita na maximální intenzitu.

`clipMap()` je pomocná funkce, která převadí jasové hodnty menší než 0 na 0 a větší než 1 na 1.

```
k1= 0.2; % zvýšení jasu
k2 = -0.2; % snížení jasu


s1 = clipMap(r + k1,0,1);
s2 = clipMap(r + k2,0,1);

figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s1);
plot(r,s2);
legend('r',['r+' num2str(k1)],['r' num2str(k2)],'Location','northwest');
```



Aplikace změny jasu na každý pixel

Využijeme maticových počtů.

`f` je matice intenzit obrázku. Je typu `uint8`, hodnoty v matici jsou od 0 (černá barva) do 255 (bílá barva).

Pokud bychom měli matici typu `double`, pak by se obrázek reprezentovaný touto maticí interpretoval tak, že 0 je nejmenší hodnota (černá barva) a 1 nejvyšší (bílá barva). Pro převod obrázků z typu `uint8` na `double` (a naopak) používáme funkci `im2double()` (respektive `im2uint8()`). Pouhé přetypování nemění hodnoty v matici. V případě `im2double()` dojde k přeškálování tak, že hodnoty větší rovny jedné se nahradí hodnotou 255, hodnoty mení rovny 0 se nahradí 0 a hodnoty mezi tím se pravidelně namapují na celočíselné hodnoty mezi tím.

```
f = imread('lena_gray.png');
jas = 33
```

```
jas =
33
```

```
tic();
g = f + 50;
toc()
```

```
Elapsed time is 0.002903 seconds.
```

```
figure,
subplot(1,2,1)
imshow(f)
title('Original')
subplot(1,2,2)
imshow(g)
title(['k = ' num2str(jas)]);
```

Funkce `tic()` a `toc()`, můžeme použít ke změření doby běhu.

Ořezávat hodnoty menší než 0 a větší než 255 není potřeba. Výsledná matice je také typu `uint8` a k ořezání dojde automaticky.

Aplikace transformace na každý pixel pomocí cyklu

V tomto případě nepoužíváme součet matice a hodnoty. V cyklu procházíme jednotlivé pixely (celý obraz) a pro každý pixel počítáme novou jasovou hodnotu.

```
%tic();
h = uint8(zeros(size(f)));
for i = 1 : size(f,1)
    for j = 1 : size(f,2)
        h(i,j) = f(i,j) + jas;
    end
end
%toc()
```

```
figure,
subplot(1,2,1)
imshow(f)
title('Original')
subplot(1,2,2)
imshow(h)
title(['k = ' num2str(jas)])
```

zeros() vrací matici typu double, je nutné ji v našem případě přetypovat na `uint8`.

Odkomentováním funkcí `tic()` a `toc()` si můžeme ověřit, že průchod obrazu pixel po pixelu je pomalejší, než využití maticového počtu.

Využití lookup tabulky (palety)

Při aplikaci jasové transformace, jak bylo výše zmíněno, dochází k mnoha zbytečným výpočtům. Všechny pixely se stejnou vstupní intenzitou budou míte stejnou výstupní intenzitu. V praxi se vypočítají nové hodnoty pro všechny jasové hodnoty a tímto způsobem získáme takzvanou lookup tabulku (paletu). Jak aplikovat paletu na obrázek jsme si ukázali první cvičení.

```
% pomocna funkce pro vytvoreni palety
map1 = createMap(s1);
map2 = createMap(s2);

% Aplikace palety na obrazek
figure
```

```
subplot(1,3,1)
imshow(f)
title('Original')
subplot(1,3,2)
imshow(f,map1)
title(['k = ' num2str(k1)]);
subplot(1,3,3)
imshow(f,map2)
title(['k = ' num2str(k2)]);
```

Aplikace palety na obrázek v matlabu nezabírá žádný čas, takže pokud chceme srovnat dobu výpočtu, stačí nám zjistit dobu potřebnou pro výpočet palety.

```
tic();
x = createMap(s1);
toc()
```

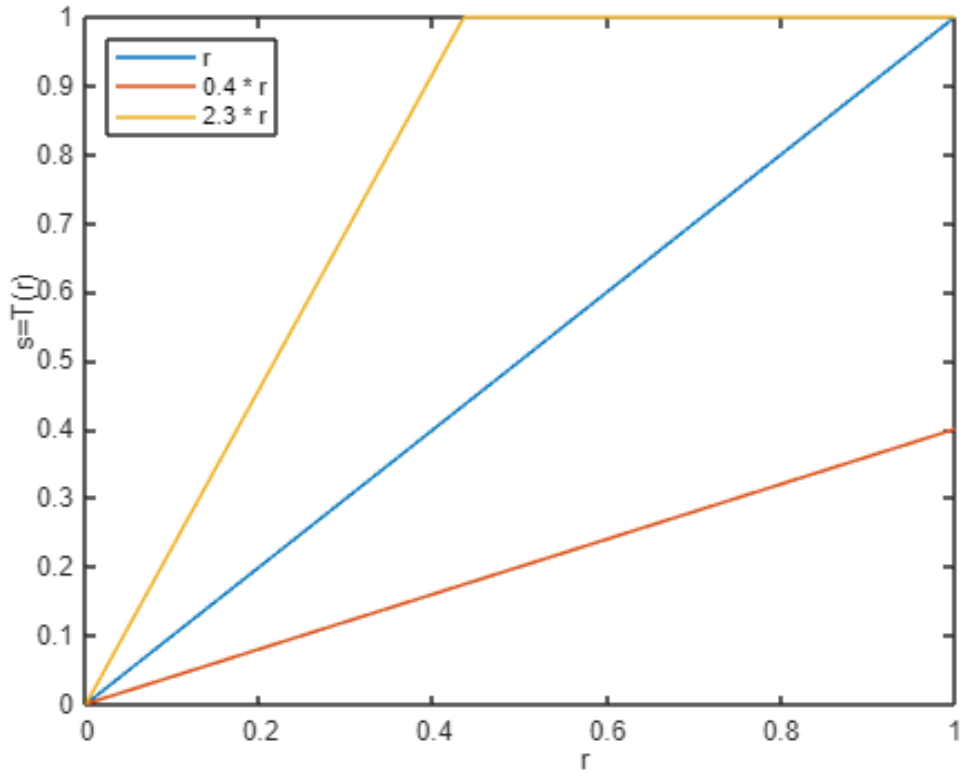
```
Elapsed time is 0.002656 seconds.
```

Změna kontrastu

Transformační funkce je ve tvaru $s = k \cdot r$, kde *k* je konstanta určující zda kontrast snižujeme, nebo zvyšujeme a jak moc.

```
c1 = 0.4;
c2 = 2.3;
```

```
s3 = clipMap(c1*r,0,1);
s4 = clipMap(c2*r,0,1);

figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s3);
plot(r,s4);
legend('r',[num2str(c1) ' * r'],[num2str(c2) ' * r'],'Location','northwest');
```



V praxi se změna kontrastu kombinuje spolu se změnou jasu.

Aplikace změny kontrastu

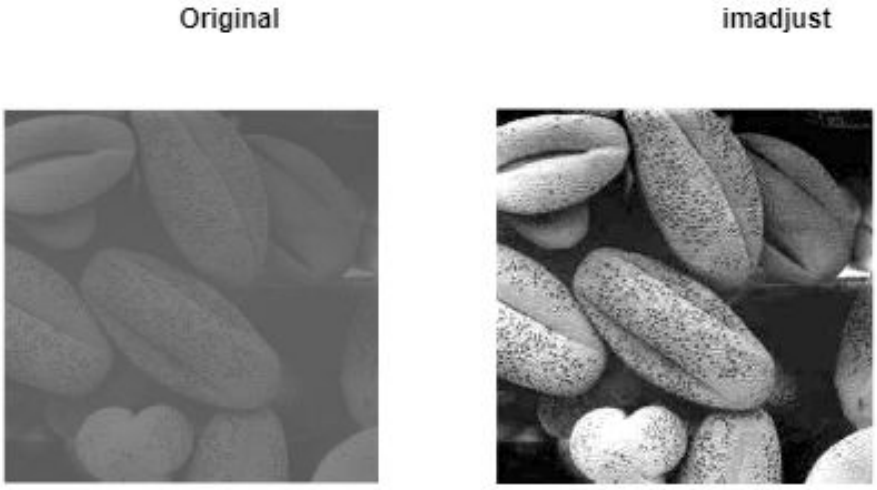
```
map3 = createMap(s3);
map4 = createMap(s4);

figure
subplot(1,3,1)
imshow(f)
title('Original')
subplot(1,3,2)
```

```
imshow(f,map3)
title(['c = ' num2str(c1)])
subplot(1,3,3)
imshow(f,map4)
title(['c = ' num2str(c2)])
```

Roztažení kontrastu

Transformační funkce zvětšení kontrastu tak, že se hodnoty v obraze roztáhnou na celý možný interval hodnot. V matlabu je tato transformace implementována ve funkci `imadjust()`.

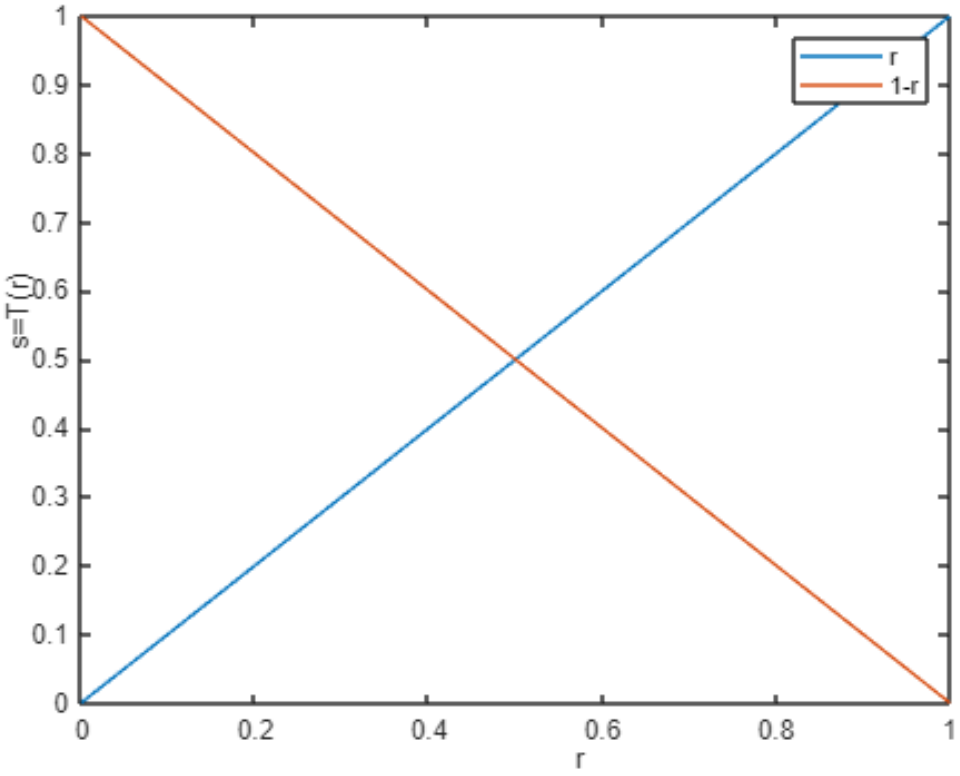
```
I = imread('pic1.png');
J = imadjust(I);

figure
subplot(1,2,1)
imshow(I);
title('Original')
subplot(1,2,2)
imshow(J);
title('imadjust')
```



Negativ obrázku

Jasová transformace má tvar $s = 1 - r$.

```
s5 = 1-r;
figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s5);
legend('r','1-r','Location','northeast');
```




```
map5 = createMap(s5);

figure
subplot(1,2,1)
imshow(f)
title('Original')
subplot(1,2,2)
imshow(f,map5)
title('Negativ')
```

Original

Negativ

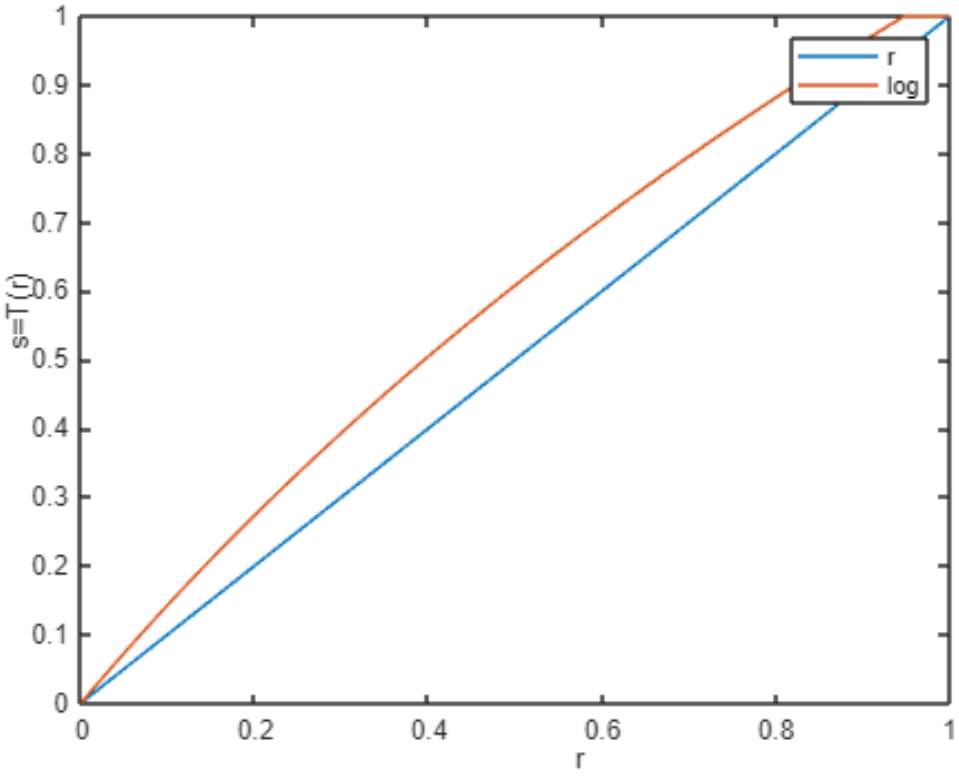
Případně můžeme využít vestavěné funkce `imcomplement()`.

```
neg = imcomplement(f);
```

Logaritmická transformace

$s = c \cdot \log(1 + r)$

```
c = 1.5;
s6 = clipMap(c*log(1+r),0,1);

figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s6);
legend('r','log','Location','northeast');
```



Logaritmická transformace se nejčastěji používá ve spojení s převodem obrázku do frekvenční domény. Příkazům není v tuto chvíli potřeba rozumět, k frekvenční doméně se dostaneme později.

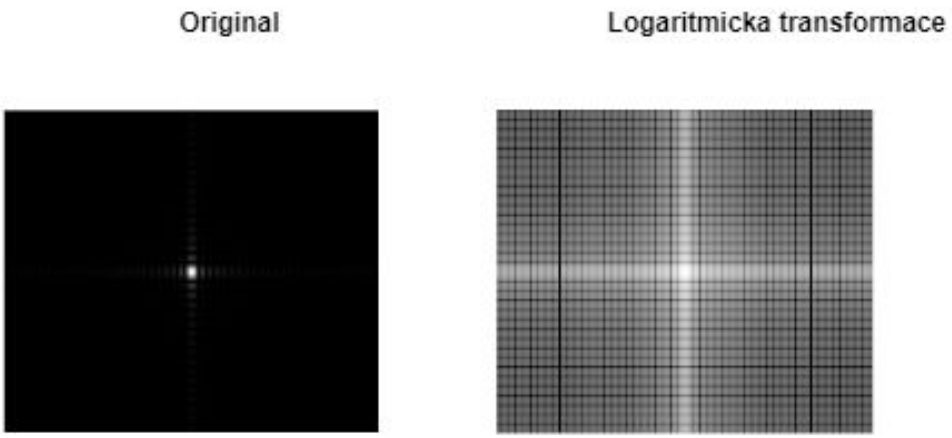
```
obrazek = rgb2gray(imread('fourier.png'));
F = fft2(obrazek);
S = abs(F);
Sc = fftshift(S);
display(min(min(Sc)));
```

```
0
```

```
display(max(max(Sc)));
```

```
442170
```

```
Scl = log(1+Sc);
figure
subplot(1,2,1)
imshow(Sc,[]);
title('Original')
subplot(1,2,2)
imshow(Scl,[]);
title('Logaritmicka transformace')
```



Gamma transformace

$s = r^{\gamma}$

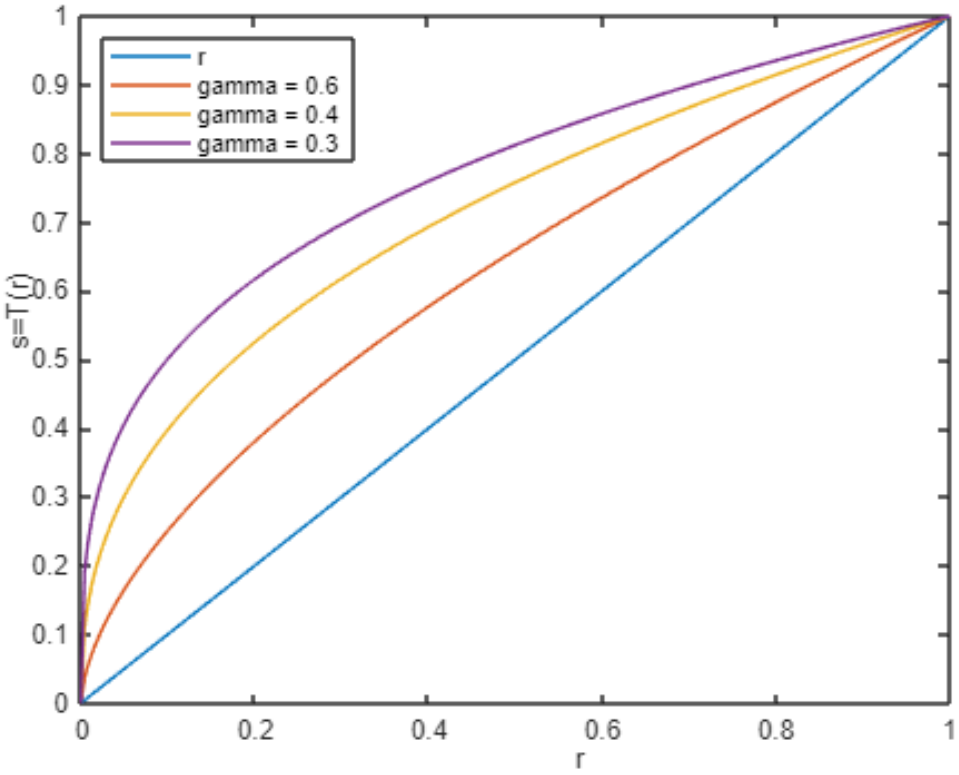
V závislosti na volbě parametru γ potlačujeme tmavou oblast a zvýrazňujeme světlou, nebo naopak.

```
c=1;
Gamma=[0.6 0.4 0.3]
```

```
Gamma = 1×3
    0.6000    0.4000    0.3000
```

```
% Prima aplikace operace na obrazek
% f=rgb2gray(imread('spine.png'));
% x1=double(x);
% y=c*(x1.^Gamma(1));
% y1=c*(x1.^Gamma(2));
% y2=c*(x1.^Gamma(3));

s7 = clipMap(c*(r.^Gamma(1)),0,1);
s8 = clipMap(c*(r.^Gamma(2)),0,1);
s9 = clipMap(c*(r.^Gamma(3)),0,1);

figure,
plot(r,r);
xlabel('r');
```

```
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s7);
plot(r,s8);
plot(r,s9);
legend('r','gamma = 0.6','gamma = 0.4','gamma = 0.3','Location','northwest');
```




```
map7 = createMap(s7);
map8 = createMap(s8);
map9 = createMap(s9);

figure
subplot(1,4,1)
imshow(f)
title('Original')
subplot(1,4,2)
imshow(f,map7)
 title('Gamma=0.6')
subplot(1,4,3)
imshow(f,map8)
 title('Gamma=0.4')
 subplot(1,4,4)
imshow(f,map9)
title('Gamma=0.3')
```

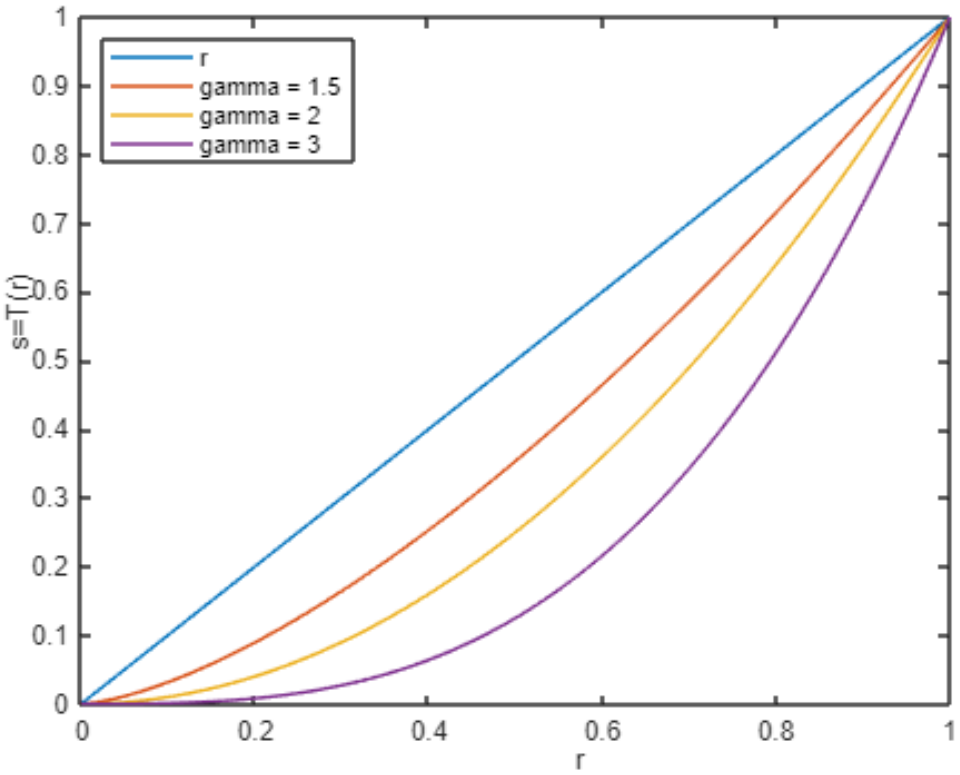
Gamma

```
c=1;
Gamma=[1.5 2 3]
```

```
Gamma = 1×3
    1.5000    2.0000    3.0000
```

```
% f = rgb2gray(imread('vase.jpg'));
% x1=double(x);
% y=c*(x1.^Gamma(1));
% y1=c*(x1.^Gamma(2));
% y2=c*(x1.^Gamma(3));

s10 = clipMap(c*(r.^Gamma(1)),0,1);
s11 = clipMap(c*(r.^Gamma(2)),0,1);
s12 = clipMap(c*(r.^Gamma(3)),0,1);

figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s10);
```

```
plot(r,s11);
plot(r,s12);
legend('r','gamma = 1.5','gamma = 2','gamma = 3','Location','northwest');
```

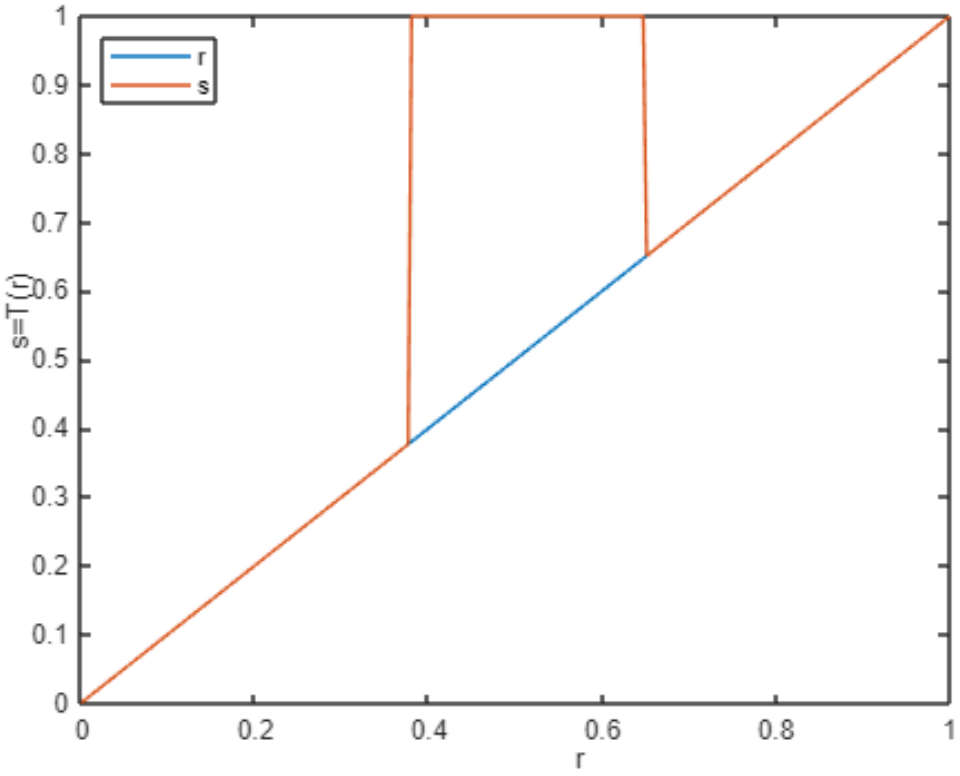



```
map10 = createMap(s10);
map11 = createMap(s11);
map12 = createMap(s12);

figure
subplot(1,4,1)
imshow(f)
title('Original')
subplot(1,4,2)
imshow(f,map10)
 title('Gamma=0.6')
subplot(1,4,3)
imshow(f,map11)
 title('Gamma=0.4')
 subplot(1,4,4)
imshow(f,map12)
title('Gamma=0.3')
```

Intensity-level slicing

Používá se ke zvýraznění některých intenzit. Ostatní intenzity se odstraní (v našem případě nastaví na bílou barvu).

```
r1 = 0.38;
r2 = 0.65;

s13 = r;
s13(s13 >= r1 & s13<=r2) = 1;

figure,
plot(r,r);
xlabel('r');
ylabel('s=T(r)');
xlim([0,1]);
hold on;
plot(r,s13);
legend('r','s','Location','northwest');
```



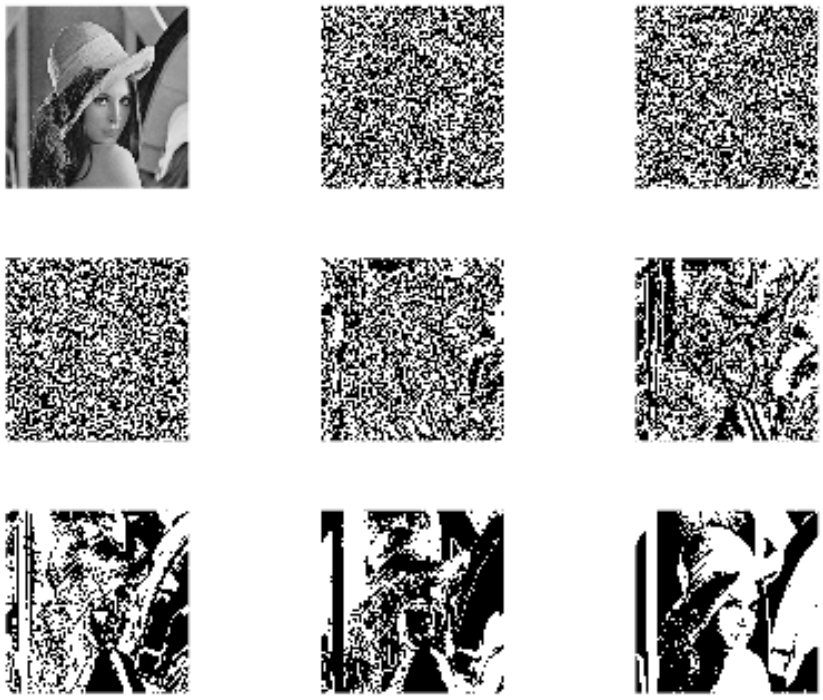

```
map13 = createMap(s13);

figure
subplot(1,2,1)
imshow(f);
title('Original')
subplot(1,2,2)
imshow(f,map13);
title('Intensity-level slicing')
```

Bit-level slicing

Slouží k ořezaní některých bitů. Například z důvodu komprese. Zde si můžeme prohlídnout jednotlivé bitové roviny.

```
I1 = bitand(f,1);
I2 = bitand(f,2);
I3 = bitand(f,4);
I4 = bitand(f,8);
I5 = bitand(f,16);
I6 = bitand(f,32);
I7 = bitand(f,64);
I8 = bitand(f,128);

figure
subplot(3,3,1)
imshow(f,[])
subplot(3,3,2)
imshow(I1,[])
subplot(3,3,3)
imshow(I2,[])
subplot(3,3,4)
imshow(I3,[])
subplot(3,3,5)
imshow(I4,[])
subplot(3,3,6)
```

```
imshow(I5,[])
subplot(3,3,7)
imshow(I6,[])
subplot(3,3,8)
imshow(I7,[])
subplot(3,3,9)
imshow(I8,[])
```



Obrázky se mohou skládat jen z několika nejvýznamnějších bitů.

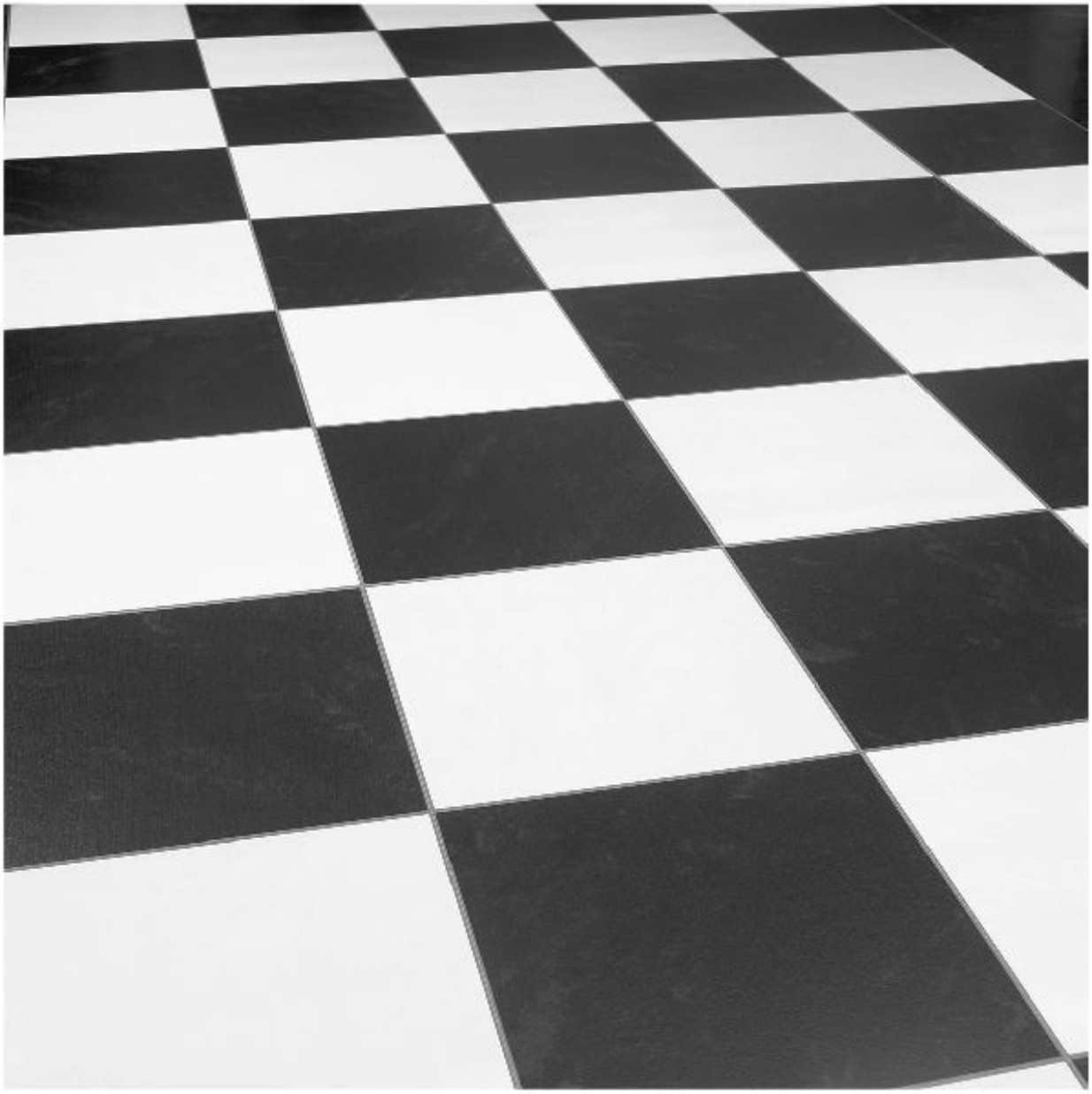
```
J1 = I8 + I7;
J2 = I8 + I7 +I6;
J3 = I8 + I7 +I6 + I5;

figure,
subplot(1,3,1)
imshow(J1);
title('Dva nejvice vyznamne bity')
subplot(1,3,2)
imshow(J2);
title('Tri nejvice vyznamne bity')
subplot(1,3,3)
imshow(J3);
title('Ctyri nejvice vyznamne bity')
```

Dva nejvice vyznamne bity Tri nejvice vyznamne bity Ctyri nejvice vyznamne b

Prahování

$$s = \begin{cases} 0, & \text{pokud } r \leq prah \\ 1, & \text{jinak} \end{cases}$$

```
I = rgb2gray(imread('chess.jpg'));
figure, imshow(I);
```



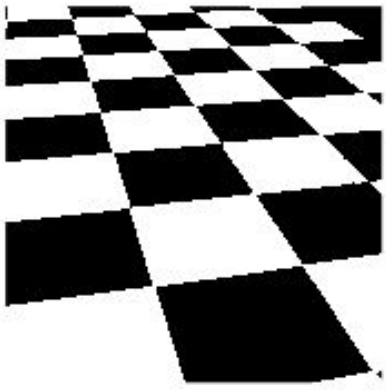
Hledání prahu - experimentálně

(více o prahování příště)

Pokud známe prahovou hodnotu, případně odhadujeme tuto hodnotu. K prahování použijeme funkci `imbinarize(obraz,prah)`. Práh se zadává v rozsahu od 0 do 1.

```
prah = 114;

J = imbinarize(I,prah/255);
% J = I > prah;
```

```
figure,
subplot(1,2,1)
imshow(I);
subplot(1,2,2)
imshow(J);
```



Úkol 2

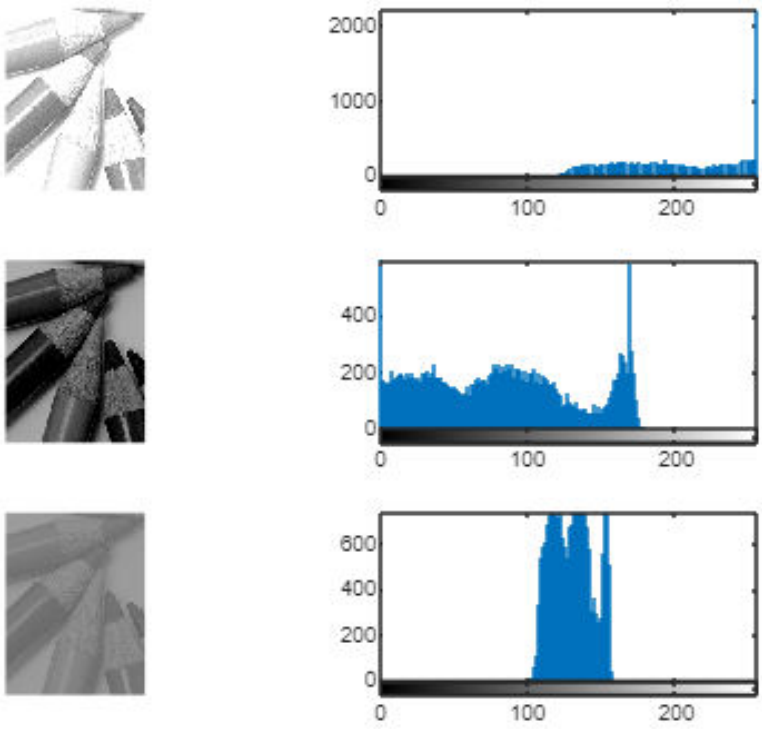
Popište, jakým způsobem se změní histogram pokud na obrázek aplikujeme následující operace. Svou doměnku ověřte pomocí matlabu.

- *změna jasu (snížení/zvýšení)*
- *změna kontrastu (snížení/zvýšení)*
- *negativ obrázku*
- *gamma korekce (pro různé gamma)*
- *prahování*

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.

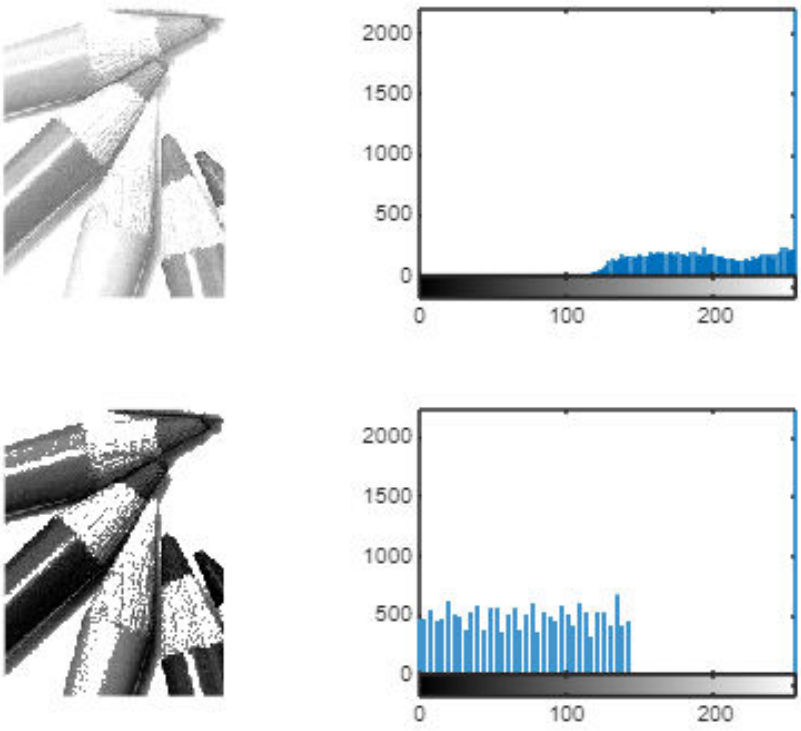
Transformace založené na histogramu

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()`.

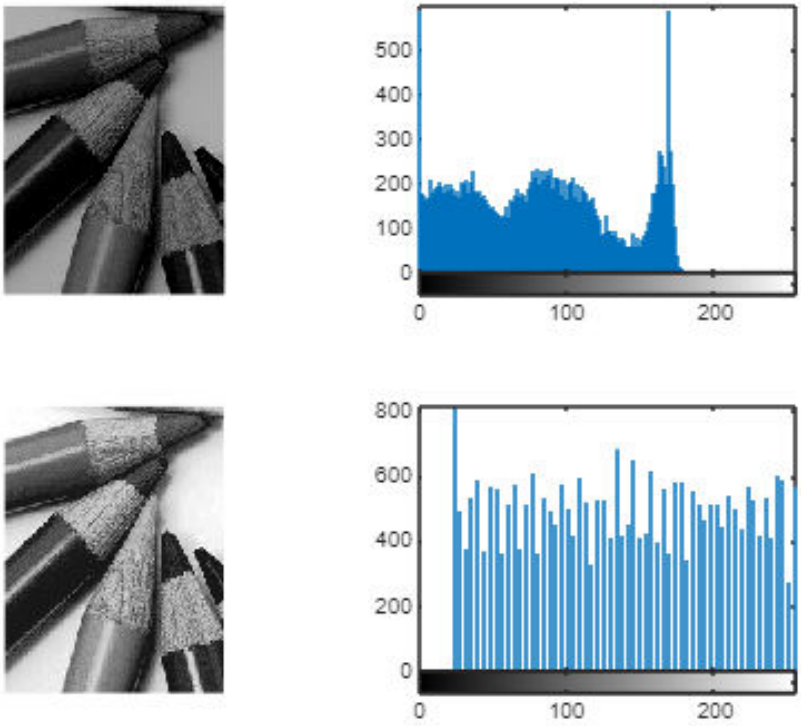
Obrazek 1

```
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)
```



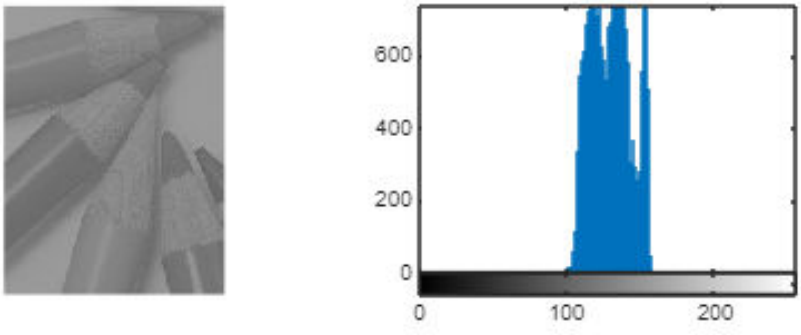
Obrázek 2

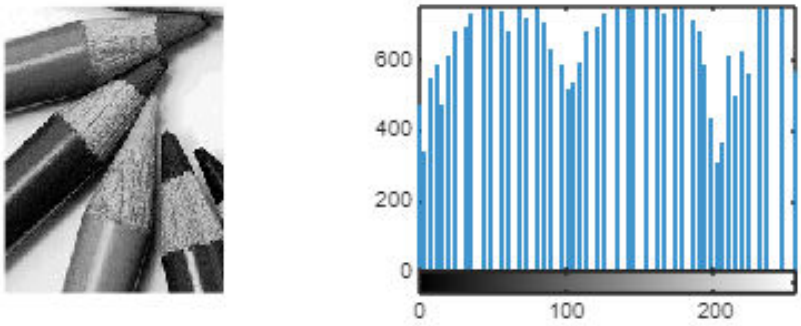
```
figure,
subplot(2,2,1)
imshow(I2);
subplot(2,2,2)
imhist(I2)
subplot(2,2,3)
imshow(J2);
subplot(2,2,4)
imhist(J2)
```



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)
```





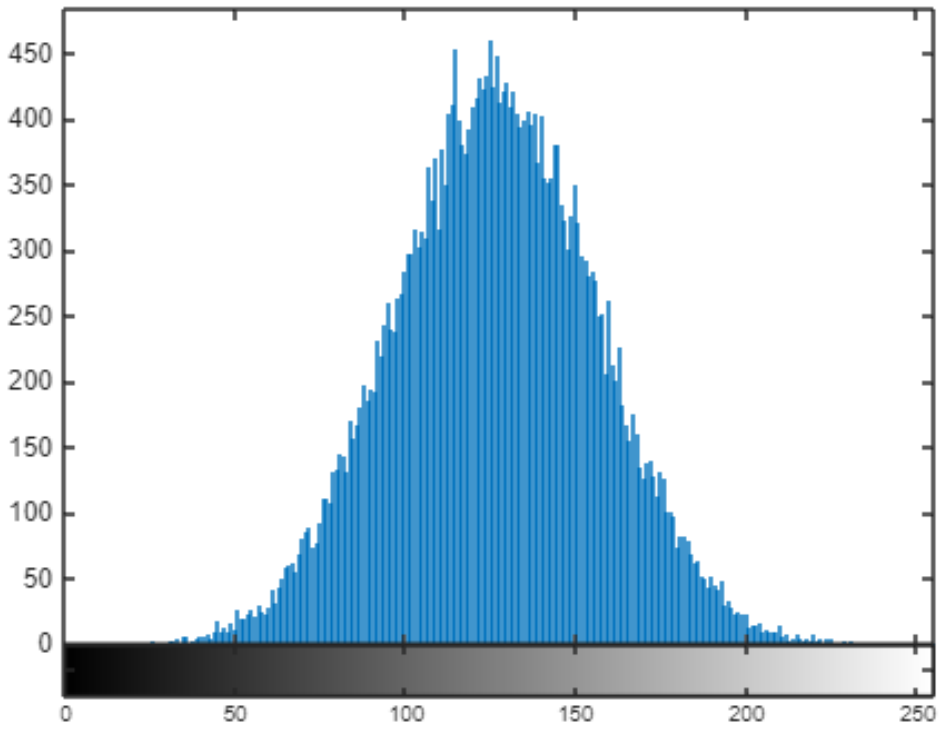
Specifikace histogramu

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.

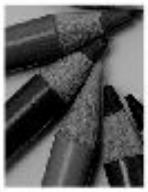
V následujícím příkladu chceme, aby výsledný tvar histogramu byl gausovská křivka (tu získáme pomocí funkce rand, která vrací náhodně hodnoty s gausovským rozložením).

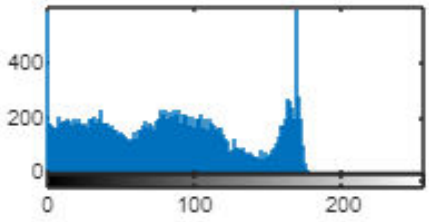
```
spechistobr = randn(size(I2));
minimum = min(min(spechistobr));
spechistobr = spechistobr + (0-minimum);
maximum = max(max(spechistobr));
spechistobr = 255*(spechistobr/maximum);
spechistobr = uint8(round(spechistobr));

figure,
imhist(spechistobr)
```

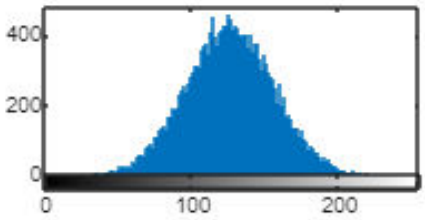


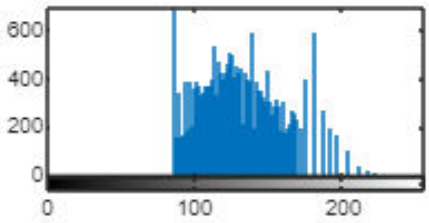
Pro operaci specifikace histogramu opět využijeme funkci `histeq()`, které jako druhý argument předáme histogram, ke kterému se chceme přiblížit.

```
[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)
```




400
200
0
0
100
200


400
200
0
0
100
200


600
400
200
0
0
100
200