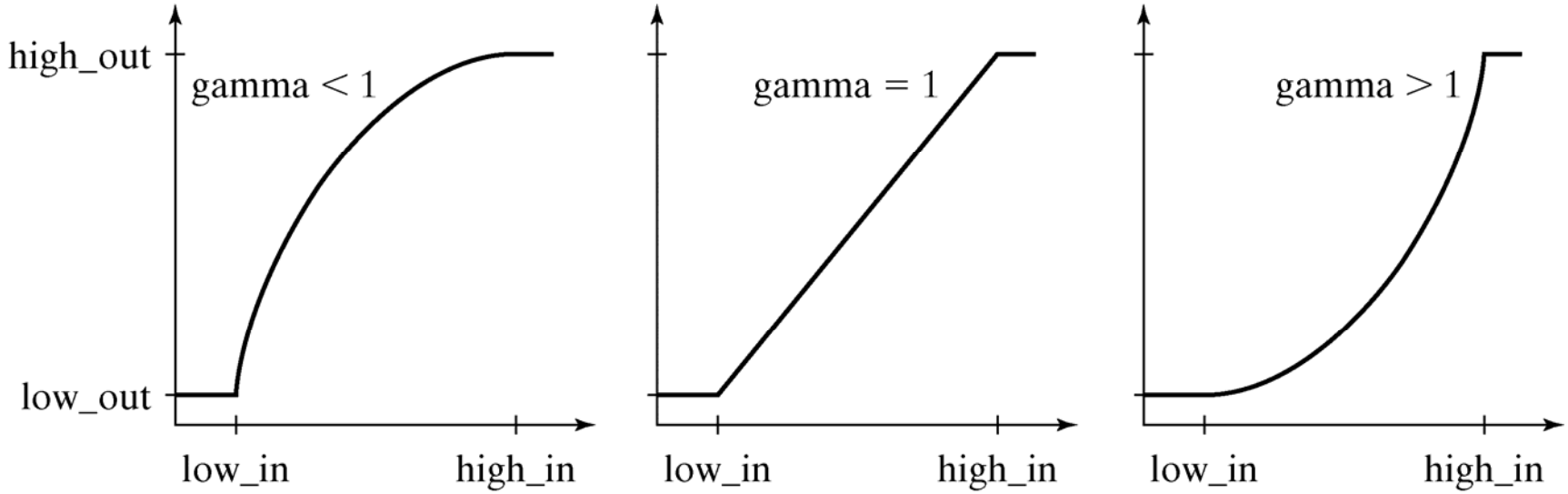

# DIGITALNA OBRADA SLIKE

## ČAS 3 – POBOLJŠANJE KVALITETA SLIKE, OPERACIJE NAD POJEDINAČNIM PIKSELIMA, PROSTORNE OPERACIJE NAD GRUPOM PIKSELA

# Razvlačenje kontrasta

```
v = imadjust(u, [low_in, high_in], [low_out, high_out], gamma)
```




Granice ***low_in***, ***high_in***, ***low_out*** i ***high_out*** se specificiraju u opsegu od 0 do 1 bez obzira na opseg ulazne slike!!!

# Razvlačenje kontrasta - primer

```
I = imread('tiffany_gray.tiff');
figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

J = imadjust(I, [110 240]/255, [0 255]/255);

figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika');
```

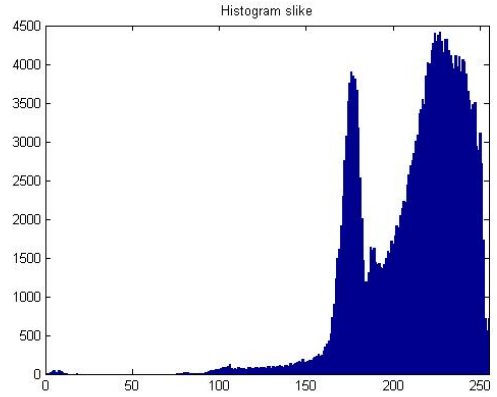
Testirati funkciju za različite vrednosti parametara. Na primer:

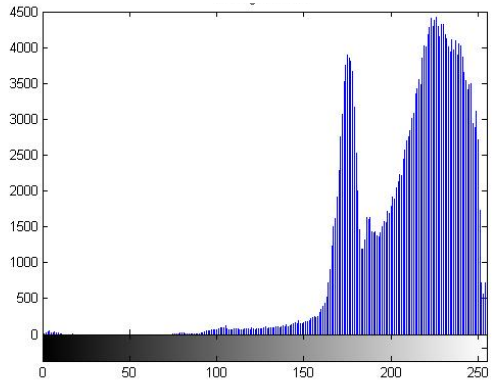
| low_in | low_out | gamma |
|---|---|---|
| 110 | 240 | 1 |
| 110 | 250 | 1 |
| 110 | 255 | 1.5 |

# Histogram slike

**h = imhist(u, broj_nivoa)**

```
I = imread('tiffany_gray.tiff');
figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

h = imhist(I);

figure; bar(h); title('Histogram slike');
set(gcf, 'Name', 'Histogram slike');
ylim('auto'); xlim([0 255]);

figure; imhist(I); title('Histogram slike');
set(gcf, 'Name', 'Histogram slike');
ylim('auto'); xlim([0 255]);
```



# Kako odrediti granice?

**k = prctile(u, p)**

```
low_in = min(I(:))
high_in = max(I(:))

J = imadjust(I, double([low_in high_in])/255, [0 255]/255);
figure; imshow(J);
set(gcf, 'Name', 'low_in = min, high_in = max');

low_in = prctile(I(:), 1)
high_in = prctile(I(:), 99)

J = imadjust(I, double([low_in high_in])/255, [0 255]/255);
figure; imshow(J);
set(gcf, 'Name', 'low_in = 1%, high_in = 99%');
```

Funkcija ***prctile*** nalazi onu vrednost niza ***u*** od koje je manje ***p%*** elementa niza ***u***. Korišćenjem ove funkcije ostvaruje se robusnost pri razvlačenju kontrasta pošto na funkciju ne utiče p% piksela sa ekstremno visokim i niskim vrednostima.

# Kako odrediti granice?

**k = stretchlim(u, tol)**
**k = stretchlim(u, [low_tol, high_tol])**

```
I = imread('dark.tif');
figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

J = imadjust(I, stretchlim(I, 0.01), [0 1]);
figure; imshow(J);
set(gcf, 'Name', 'low_in = 1%, high_in = 99%');

J = imadjust(I, stretchlim(I, [0.12, 0.99]), [0 1]);
figure; imshow(J);
set(gcf, 'Name', 'low_in = 12%, high_in = 99%');
```

Funkcija ***stretchlim*** određuje gornju i donju granicu za razvlačenje kontrasta tako da se zasiti ***low_tol*100%*** najtamnijih piksela i ***high_tol*100%*** najsvetlijih piksela ulazne slike.

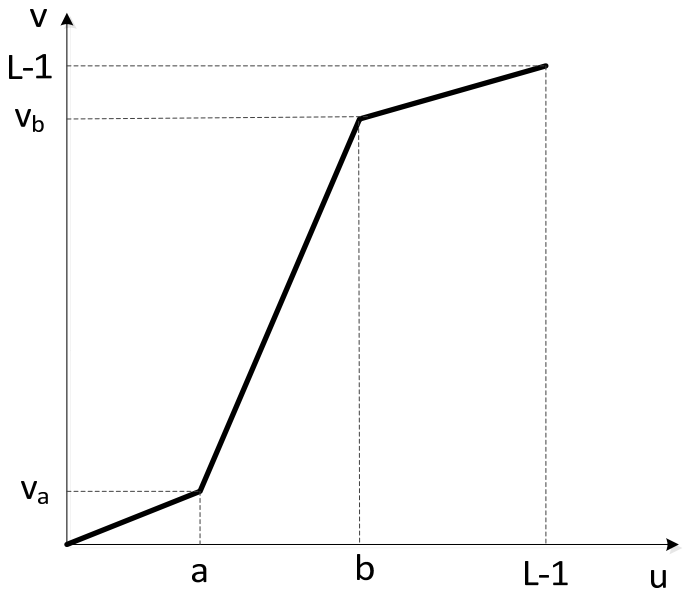
Kada se poziva samo sa jednim argumentom onda važi:

***low_tol = tol***
***high_tol = 1 - tol***

# Proizvoljna funkcija za razvlačenje kontrasta

**v = interp1(x_val, y_val, u)**

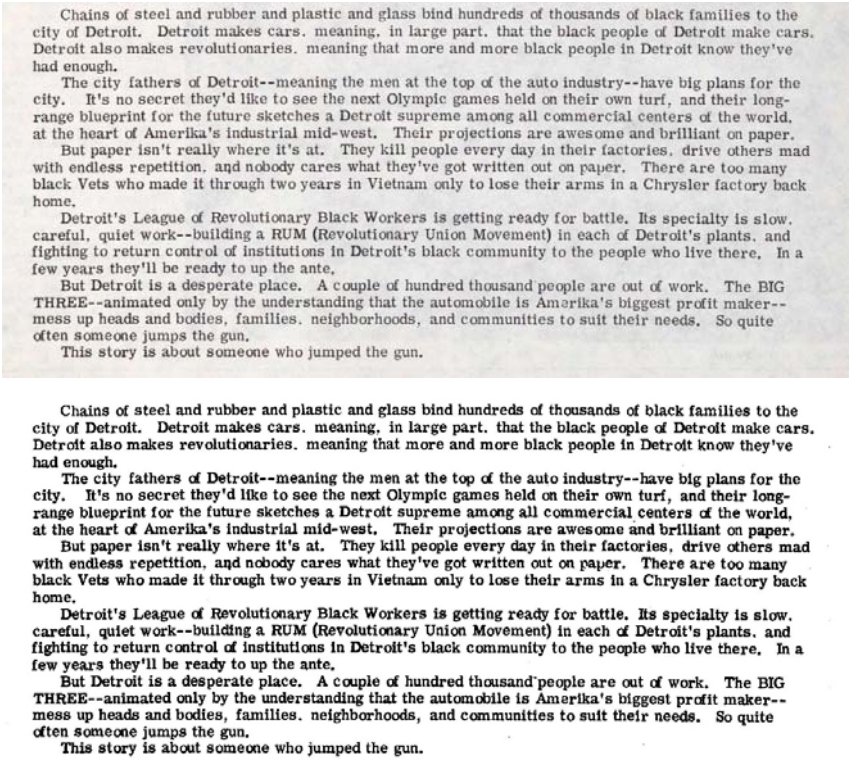
```
I = imread('tiffany_gray.tiff');

a = 110; b = 230; va = 20; vb = 240;

x_val = [0 a b 255];
y_val = [0 va vb 255];

J = uint8(interp1(x_val, y_val, double(I)));

figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika');
```



# Poređenje s pragom (binarizacija)

**v = im2bw(u, prag)**

```
I = imread('text.jpg');
figure; imshow(I);
set(gcf, 'Name', 'Skenirana slika teksta');

I = rgb2gray(I);
figure; imhist(I); ylim('auto');
title('Histogram skeniranog teksta');

J = 255*uint8(im2bw(I, 200./255));

figure; imshow(J);
set(gcf, 'Name', 'Binarna slika teksta');
```



Chains of steel and rubber and plastic and glass bind hundreds of thousands of black families to the city of Detroit. Detroit makes cars, meaning, in large part, that the black people of Detroit make cars. Detroit also makes revolutionaries, meaning that more and more black people in Detroit know they've had enough.

The city fathers of Detroit--meaning the men at the top of the auto industry--have big plans for the city. It's no secret they'd like to see the next Olympic games held on their own turf, and their long-range blueprint for the future sketches a Detroit supreme among all commercial centers of the world, at the heart of Amerika's industrial mid-west. Their projections are awesome and brilliant on paper.

But paper isn't really where it's at. They kill people every day in their factories, drive others mad with endless repetition, and nobody cares what they've got written out on paper. There are too many black Vets who made it through two years in Vietnam only to lose their arms in a Chrysler factory back home.

Detroit's League of Revolutionary Black Workers is getting ready for battle. Its specialty is slow, careful, quiet work--building a RUM (Revolutionary Union Movement) in each of Detroit's plants, and fighting to return control of institutions in Detroit's black community to the people who live there. In a few years they'll be ready to up the ante.

But Detroit is a desperate place. A couple of hundred thousand people are out of work. The BIG THREE--animated only by the understanding that the automobile is Amerika's biggest profit maker--mess up heads and bodies, families, neighborhoods, and communities to suit their needs. So quite often someone jumps the gun.

This story is about someone who jumped the gun.

# Kako odrediti prag?

```
I = imread('text.jpg');
I = im2double(rgb2gray(I));

t = 0.5;
tn = (mean(I(I<=t)) + mean(I(I>t)))/2;

while (abs(tn - t) > 0.001)
t = tn;
tn = (mean(I(I<=t)) + mean(I(I>t)))/2;
end

t

figure; imshow(im2bw(I, t));
set(gcf, 'Name', 'Binarna slika teksta');
```

Cilj je odrediti prag koji najbolje deli histogram na dva dela. Globalni prag ima smisla u slučaju jasno razdvojenih objekata i pozadine.

Jedna od ideja je da se na početku usvoji proizvoljna vrednost praga koja deli sliku na dve klase. Zatim se računaju srednje vrednosti elemenata u svakoj od klasa i nova poboljšana vrednost praga predstavlja aritmetičku sredinu ovih srednjih vrednosti klasa.

Nakon toga se ponavlja iteracija sve dok vrednost praga ne konvergira sa odgovarajućim stepenom tolerancije.

# Kako odrediti prag?

**t = graythresh(u)**

```
t = graythresh(I)

figure; imshow(im2bw(I, t));
set(gcf, 'Name', 'Binarna slika teksta');
```

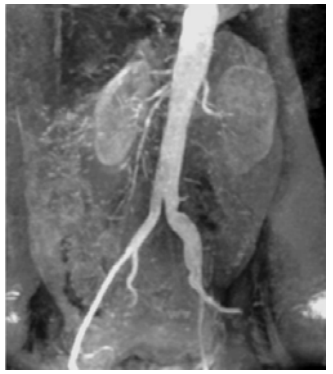
Funkcija određuje optimalni globalni prag korišćenjem Otsu metode.

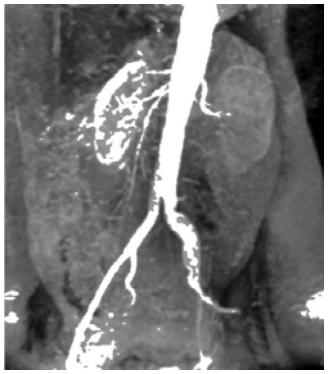
Ova metoda pored srednje vrednosti unutar svake klase, uzima u obzir i varijansu unutar svake od klasa kao i globalnu varijansu. Određeni prag je optimalan u smislu da maksimizuje varijansu između dve klase piksela.

Uporediti izlaz ove funkcije sa pragom određenim na prethodnom slajdu.

# Izdvajanje regiona zadate sjajnosti

```
I = imread('kidney.tif');

figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

J = zeros(size(I));
J((I>=155) & (I<=235)) = 255;
figure; imshow(J);
set(gcf, 'Name', 'Izdvojeni region');

J = I;
J((I>=155) & (I<=235)) = 255;
figure; imshow(J);
set(gcf, 'Name', 'Izdvojeni region sa pozadinom');
```





# Izdvajanje bit ravni

```
I = imread('dollar.tif');

Ib7 = 255*(bitand(I, 128)/128);
Ib6 = 255*(bitand(I, 64)/64);
Ib5 = 255*(bitand(I, 32)/32);
Ib4 = 255*(bitand(I, 16)/16);
Ib3 = 255*(bitand(I, 8)/8);
Ib2 = 255*(bitand(I, 4)/4);
Ib1 = 255*(bitand(I, 2)/2);
Ib0 = 255*(bitand(I, 1));

J1 = 128*(Ib7/255) + 64*(Ib6/255);
J2 = J1 + 32*(Ib5/255);
J3 = J2 + 16*(Ib4/255);
```

**v = bitand(u, maska)**

**I**

**J1**

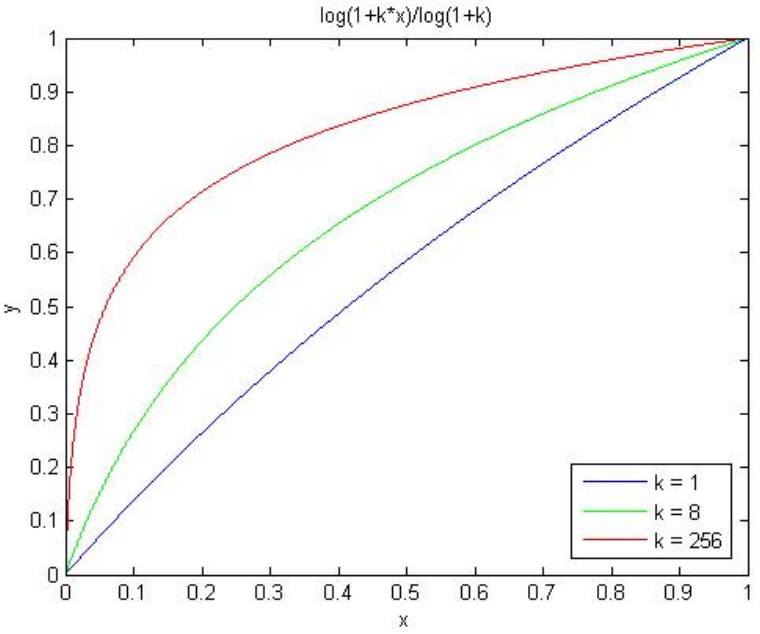
**J2**

**J3**

# Kompresija kontrasta – log funkcija

**v = c*log(1+k*u)**

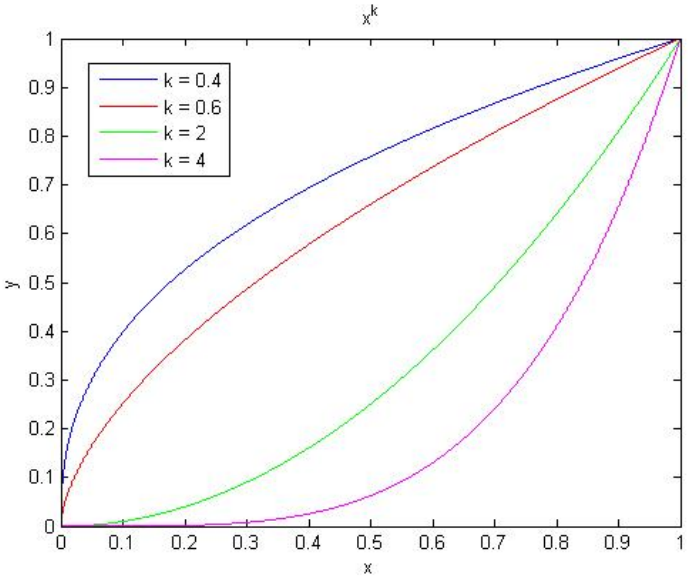
```
I = hdrread('belgium.hdr');

I = rgb2gray(I);
I = I./max(I(:));
figure; imshow(I);
set(gcf, 'Name', 'Linearno skalirana slika');

J = log(1 + 256*I)/log(257);
figure; imshow(J);
set(gcf, 'Name', 'Izlaz posle log funkcije za k = 256');

J = log(1 + 3096*I)/log(3097);
figure; imshow(J);
set(gcf, 'Name', 'Izlaz posle log funkcije za k = 3096');
```



# Kompresija kontrasta – stepena funkcija

**v = c*u.^k**

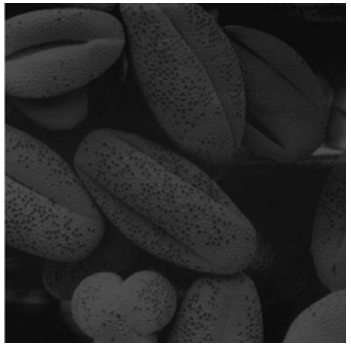
```
I = hdrread('belgium.hdr');

I = rgb2gray(I);
I = I./max(I(:));
figure; imshow(I);
set(gcf, 'Name', 'Linearno skalirana slika');

J = I.^0.4;
figure; imshow(J);
set(gcf, 'Name', 'Izlaz posle stepene funkcije za k = 0.4');

J = I.^0.22;
figure; imshow(J);
set(gcf, 'Name', 'Izlaz posle stepene funkcije za k = 0.22');
```

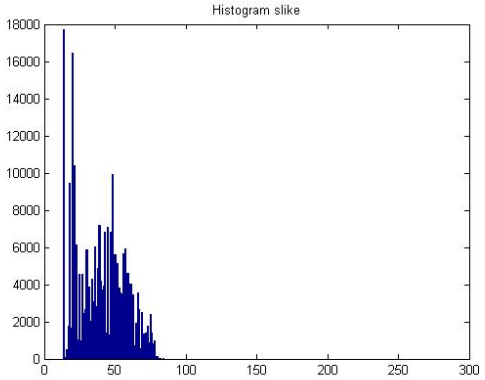


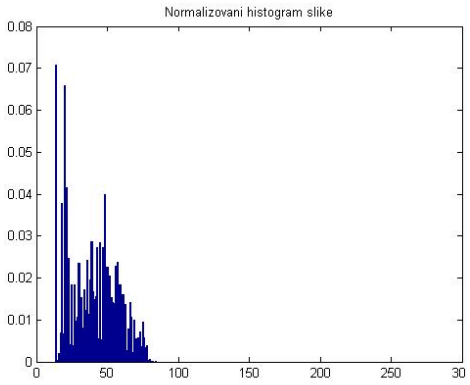

**Poboljšati kvalitet slike areal.tif!**

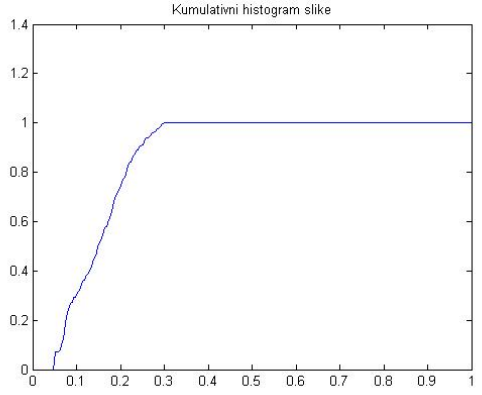
# Histogram slike

```
I = imread('dark.tif');

h = imhist(I);
figure; bar(h);
title('Histogram slike');

hnorm = h./numel(I);
figure; bar(hnorm);
title('Normalizovani histogram slike');

cdf = cumsum(hnorm);
x = linspace(0,1,256);
figure; plot(x, cdf);
title('Kumulativni histogram slike');
```





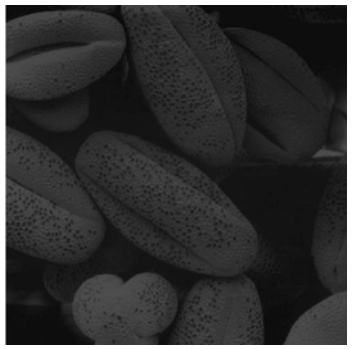



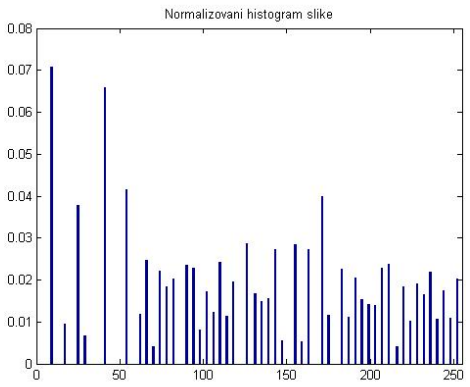



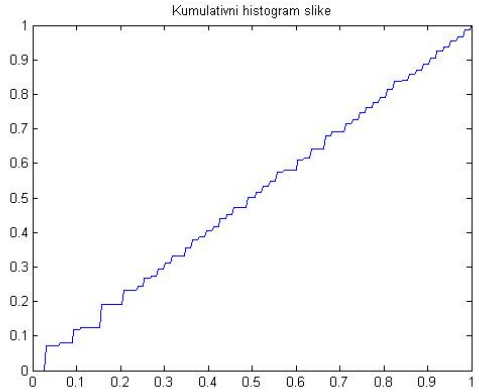
# Ekvalizacija histograma

**v = histeq(u, broj_nivoa)**

```
I = imread('dark.tif');

J = histeq(I);

hnorm = imhist(J)./numel(J);
figure; bar(hnorm);
ylim('auto'); xlim([0 255]);
title('Normalizovani histogram slike');

cdf = cumsum(hnorm);
x = linspace(0,1,256);
figure; plot(x, cdf); xlim([0 1]);
title('Kumulativni histogram slike');
```

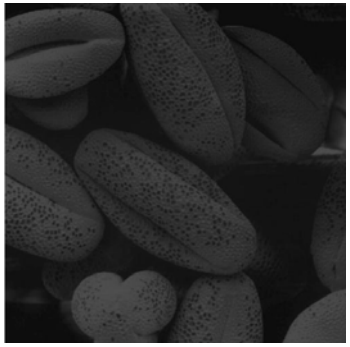


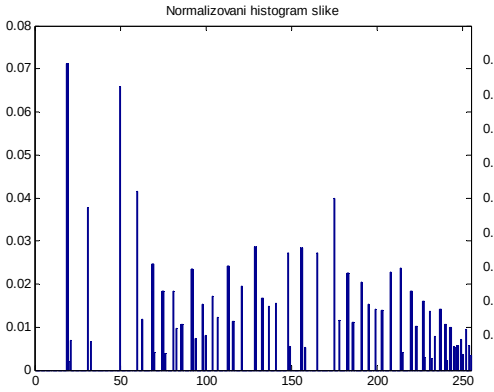


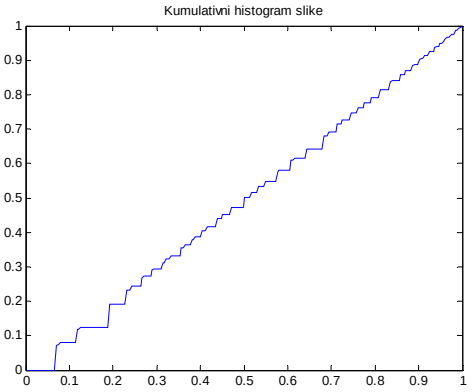



# Ekvalizacija histograma – bez histeq

```
I = imread('dark.tif');

hnorm = imhist(I)./numel(I);
cdf = cumsum(hnorm);

lut = uint8(round(cdf*255));
J = intlut(I, lut); figure; imshow(J);

hnorm = imhist(J)./numel(J);
figure; bar(hnorm);
ylim('auto'); xlim([0 255]);
title('Normalizovani histogram slike');

cdf = cumsum(hnorm);
x = linspace(0,1,256);
figure; plot(x, cdf); xlim([0 1]);
title('Kumulativni histogram slike');
```



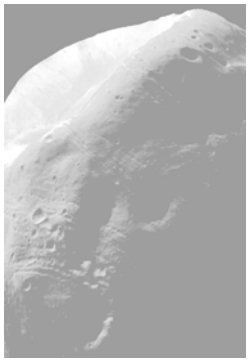


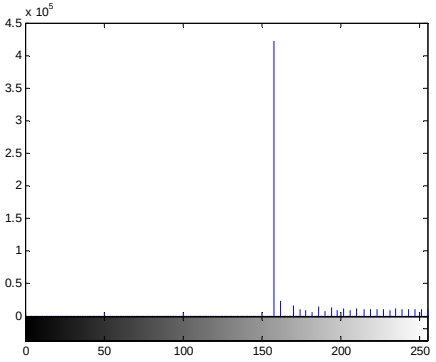



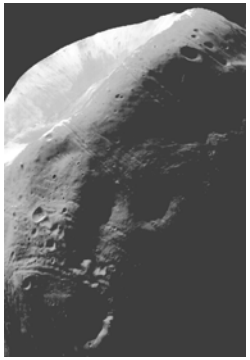
# Specifikacija histograma

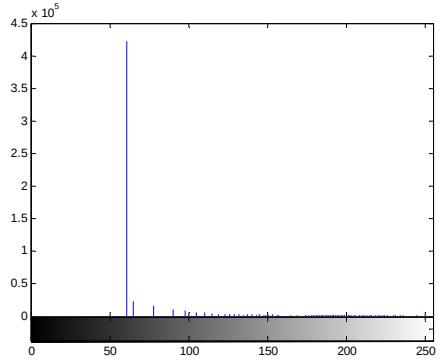
**v = histeq(u, ciljni_histogram)**

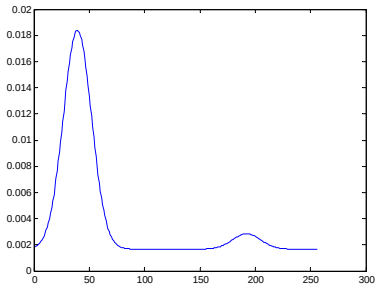
```
I = imread('mars_moon.tif');

Jhe = histeq(I);
figure; imshow(Jhe);
figure; imhist(Jhe); ylim('auto');

load target_hist
figure; plot(target_hist);
Jhs = histeq(I, target_hist);
figure; imshow(Jhs);
figure; imhist(Jhs); ylim('auto');
```







# Adaptivna ekvalizacija histograma

**v = adapthisteq(u)**

```
I = imread('mars_moon.tif');

J = adapthisteq(I, 'ClipLimit', 0.08,...
 'NumTiles', [8 4]);

figure; imshow(J);
```

Koristi se CLAHE (Contrast Limited Adaptive Histogram Equalization) algoritam.

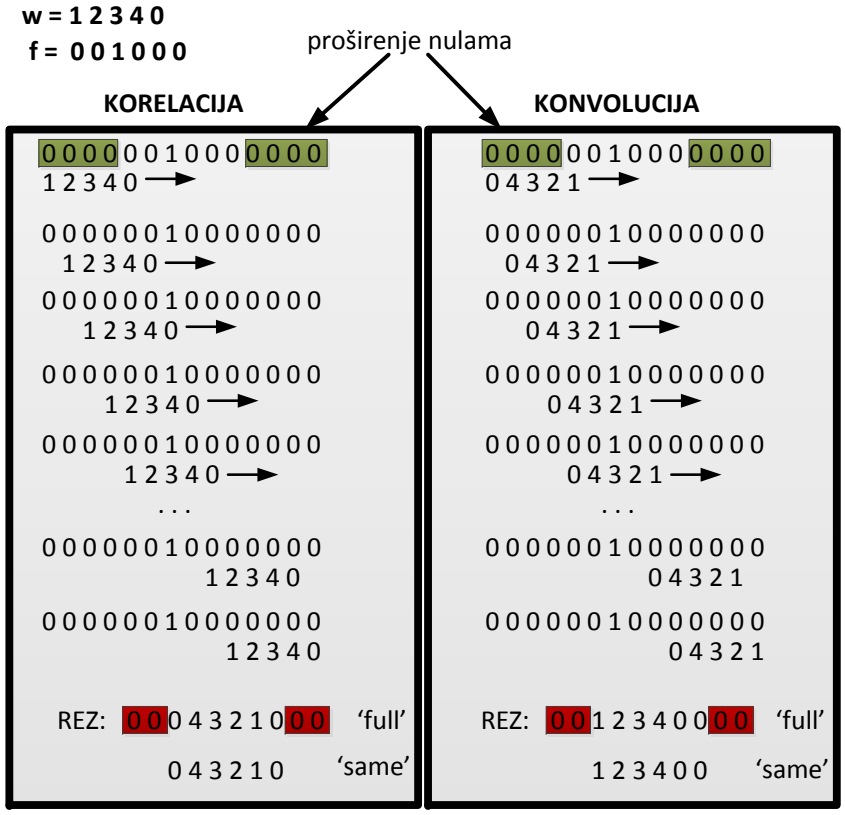
Parametri algoritma su:

'ClipLimit' – granica odsecanja histograma

'NumTiles' – niz sa dva elementa određuje broj blokova po horizontali i vertikali

# PROSTORNE OPERACIJE NAD GRUPOM PIKSELA

# Korelacija vs konvolucija




**Korealcija** $$y[m,n] = \sum_{k=-a}^{a} \sum_{l=-b}^{b} w[k,l]x[m+k,n+l]$$

**Konvolucija** $$y[m,n] = \sum_{k=-a}^{a} \sum_{l=-b}^{b} w[k,l]x[m-k,n-l]$$

**Postupak specificiranja prostorne maske i mehanizam filtriranja podrazumevaju da se obavlja korelacija. U slučaju konvolucije potrebno je zarotirati prostornu masku.**

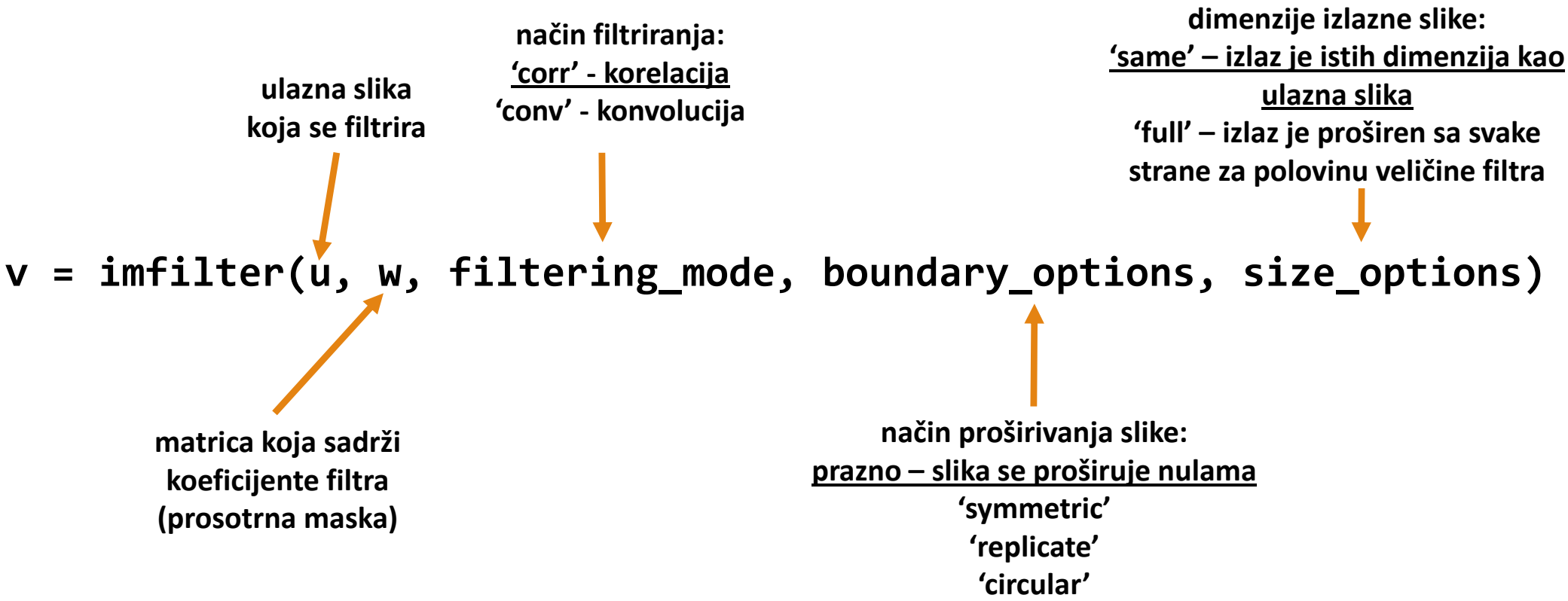
**Korealcija**

$$w = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

**Konvolucija**

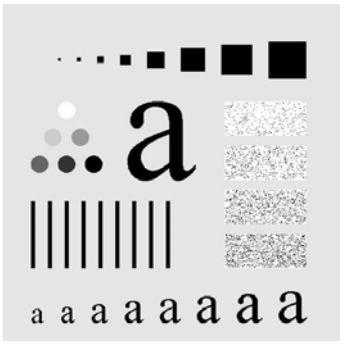
$$w = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

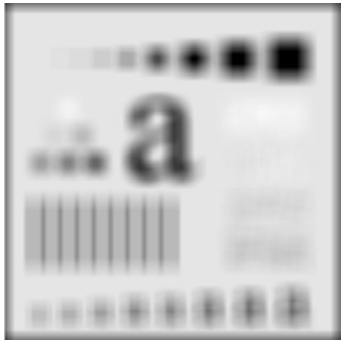
# Filtriranje u prostornom domenu



# Prostorno usrednjavanje

```
I = imread('test_pattern.tif');
w = ones(31,31);
figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika')
J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika - maska nije normalizovana');
w = w./sum(w(:));
J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika - normalizovana maska');
J = imfilter(I, w);
figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika - prosirivanje nulama');
```





# Generisanje prostorne maske

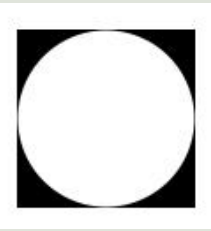
```
w = fspecial('type', parameters)
```

**`fspecial('average', [m n])`**

**specificira MxN masku za usrednjavanje sa jednakim težinama za sve piksele**

**`fspecial('disk', r)`**

**Cirkularna maska za usrednjavanje radijusa r. Dimenzije maske su (2r+1)x(2r+1)**



**`fspecial('gaussian', [m n], sigma)`**

**MxN maska za težinsko usrednjavanje. Težine raspoređenje po Gausovoj funkciji standardne devijacije sigma.**

**`fspecial('laplacian', alpha)`**

**Maska za računanje Laplasijana. Koeficijenti zavise od alpha prema izrazu:**

$$\begin{bmatrix} \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \\ \frac{1-\alpha}{1+\alpha} & \frac{-4}{1+\alpha} & \frac{1-\alpha}{1+\alpha} \\ \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \end{bmatrix}$$

# Generisanje prostorne maske

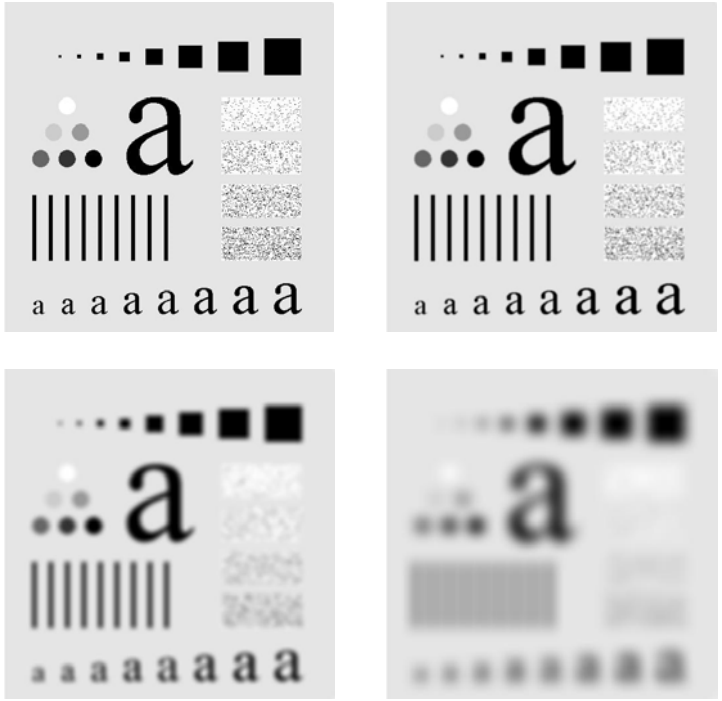
**[M, N] = meshgrid(m, n)**

```
m = -50:50;
n = -50:50;

[M, N] = meshgrid(m, n);
sigma = 25;
w = exp(-(M.^2+N.^2)/(2*sigma^2));
w = w/sum(w(:));
figure; imshow(w, []);

w1 = fspecial('gaussian', [101, 101], 25);
figure; imshow(w1, []);

max(w1(:) - w(:))
```

```
[M, N] = meshgrid(-1:1, -1:1)

M =

    -1     0     1
    -1     0     1
    -1     0     1

N =

    -1    -1    -1
     0     0     0
     1     1     1
```

# Prostorno usrednjavanje

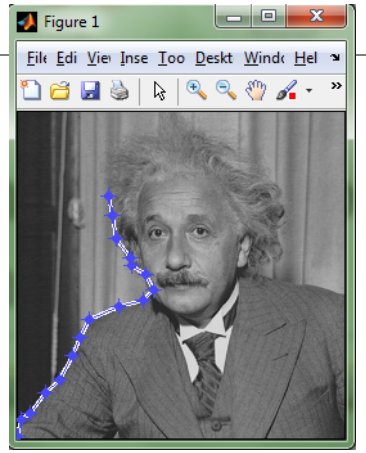
```
I = imread('test_pattern.tif');

w = fspecial('gaussian', [5 5], 1);
J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', '5x5 Gausov filtar - sigma = 1');

w = fspecial('gaussian', [15 15], 4);
J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', '15x15 Gausov filtar - sigma = 4');

w = fspecial('gaussian', [41 41], 10);
J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', '41x41 Gausov filtar - sigma = 10');
```

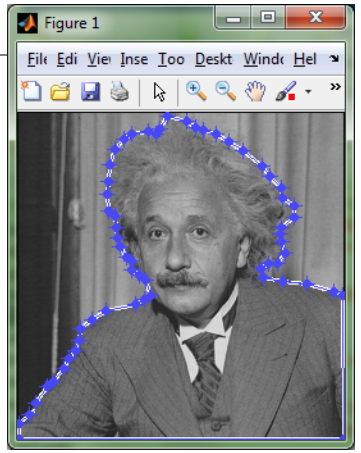


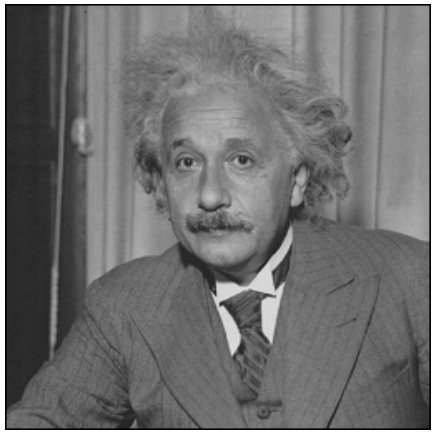
# Interaktivno određivanje regiona od interesa

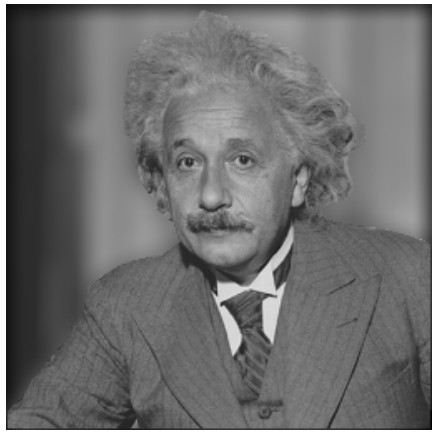
```
I = imread('einstein.tif');
mask = roipoly(I);

w_lp = fspecial('gaussian', [21 21], 7);

Ib = imfilter(I, w_lp, 'replicate');
J = Ib;
J(mask) = I(mask);
figure; imshow(J);
```

dupli klik na početnu tačku zatvara poligon



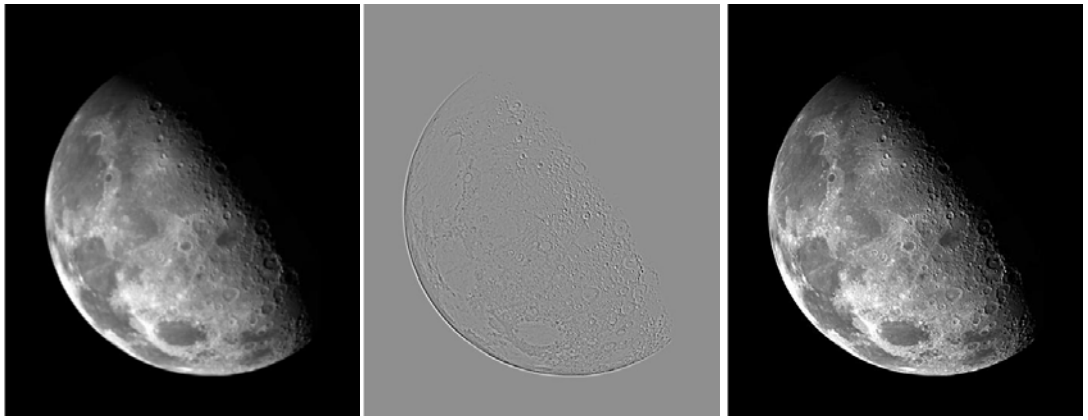





# Izoštravanje slike korišćenjem Laplasijana

```
I = imread('blurry_moon.tif');

figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

w = [0 1 0; 1 -4 1; 0 1 0];

I_lap = imfilter(double(I), w, 'replicate');
figure; imshow(I_lap, []);
set(gcf, 'Name', 'Laplasijan ulazne slike');

J = uint8(double(I) - I_lap);
figure; imshow(J);
set(gcf, 'Name', 'Izlazna izostrena slika');
```

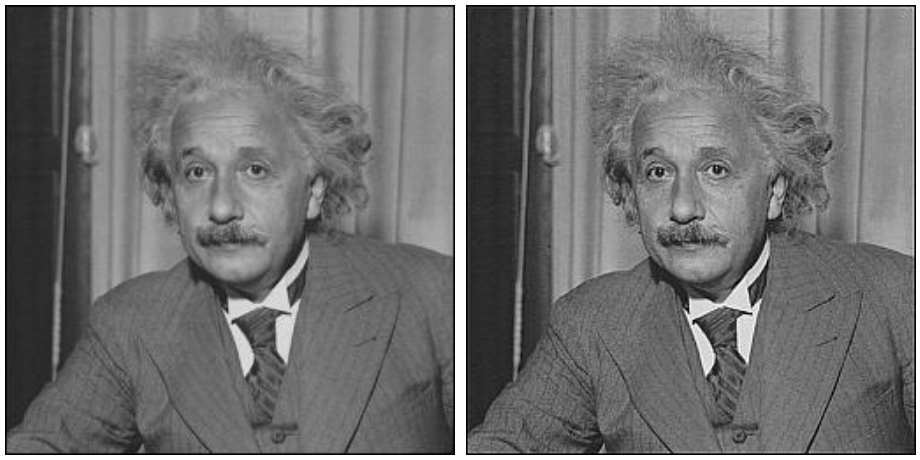


**Testirajte i drugu definiciju Laplasijana:**

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

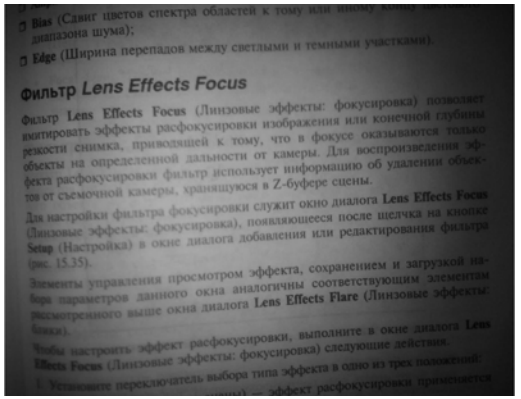
# Isticanje visokih učestanosti

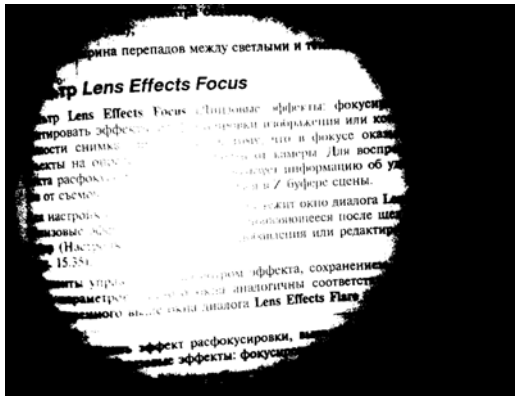
```
I = double(imread('einstein.tif'));

w_lp = fspecial('average', 3);
I_lp = imfilter(I, w_lp, 'replicate');
I_hp = I - I_lp;
J = uint8(I + I_hp);
figure; imshow(J);

w_sharp = [-1 -1 -1; -1 17 -1; -1 -1 -1]./9;
J1 = uint8(imfilter(I, w_sharp, 'replicate'));
figure; imshow(J1);

max(J(:)-J1(:))
```

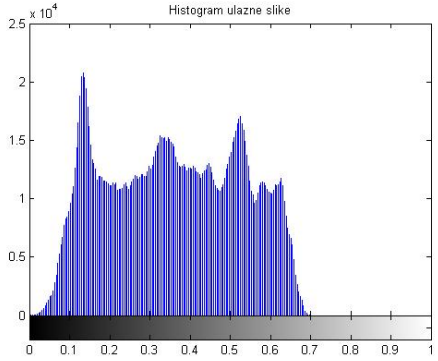


# Problematična binarizacija

```
I = imread('text-from-book-big.jpg');
I = double(rgb2gray(I))/255;

figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');
figure; imhist(I); ylim('auto');
title('Histogram ulazne slike');

Jb = im2bw(I, graythresh(I));
figure; imshow(Jb);
set(gcf, 'Name', 'Pokusaj binarizacije');
```

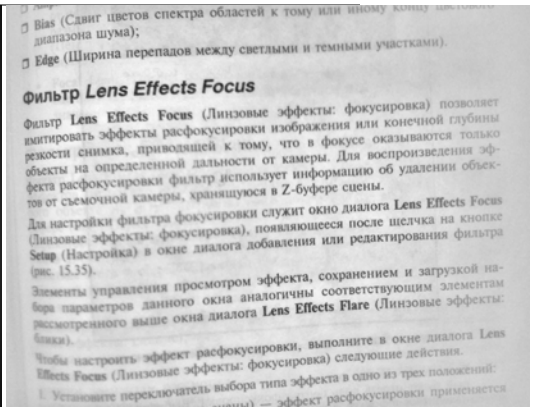


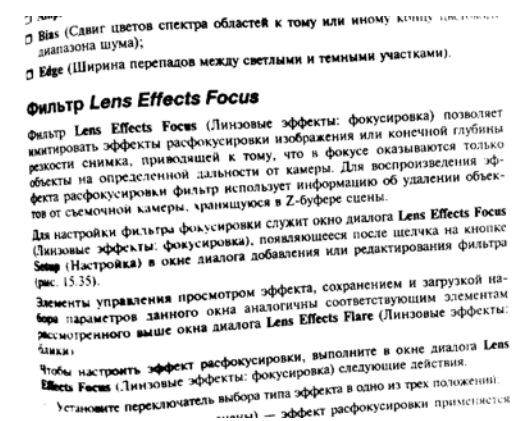




# Problematična binarizacija

```
w_lp = fspecial('gaussian', [201, 201], 60);
I_lp = imfilter(I, w_lp, 'replicate');
figure; imshow(I_lp);

J = mat2gray(I - I_lp);

figure; imshow(J);
set(gcf, 'Name', 'Slika posle potiskivanja
niskih ucestanosti');

figure; imhist(J); ylim('auto');
title('Histogram modifikovane slike');

Jb = im2bw(J, graythresh(J));
figure; imshow(Jb);
set(gcf, 'Name', 'Izlazna binarna slika');
```





Bias (Сдвиг цветов спектра областей к тому или иному
диапазона шума);

Edge (Ширина перепадов между светлыми и темными участками).

**Фильтр *Lens Effects Focus***

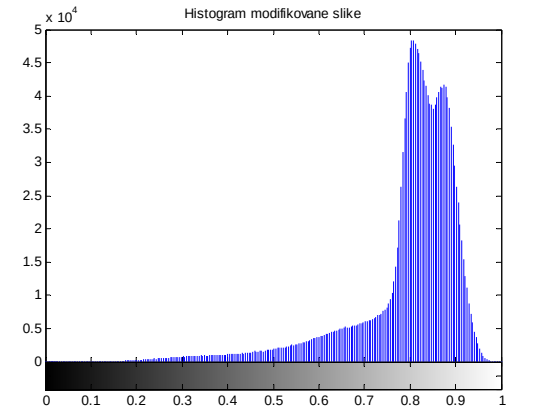
Фильтр **Lens Effects Focus** (Линзовые эффекты: фокусировка) позволяет имитировать эффекты расфокусировки изображения или конечной глубины резкости снимка, приводящей к тому, что в фокусе оказываются только объекты на определенной дальности от камеры. Для воспроизведения эффекта расфокусировки фильтр использует информацию об удалении объектов от съемочной камеры, хранящуюся в Z-буфере сцены.

Для настройки фильтра фокусировки служит окно диалога **Lens Effects Focus** (Линзовые эффекты: фокусировка), появляющееся после щелчка на кнопке **Setup** (Настройка) в окне диалога добавления или редактирования фильтра (рис. 15.35).

Элементы управления просмотром эффекта, сохранением и загрузкой набора параметров данного окна аналогичны соответствующим элементам рассмотренного выше окна диалога **Lens Effects Flare** (Линзовые эффекты: блики).

Чтобы настроить эффект расфокусировки, выполните в окне диалога **Lens Effects Focus** (Линзовые эффекты: фокусировка) следующие действия.

1. Установите переключатель выбора типа эффекта в одно из трех положений:



# Nelinearno filtriranje

```
v = medfilt2(u, [m n], border_options)
v = ordfilt2(u, redni_broj, maska)
```

```
ordfilt2(u, 1, ones(m,n))
```
**Vraća minimalni element iz pravougaone maske MxN.**

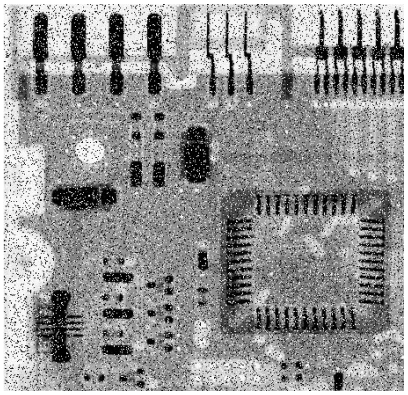
```
ordfilt2(u, m*n, ones(m,n))
```
**Vraća maskimalni element iz pravougaone maske MxN.**

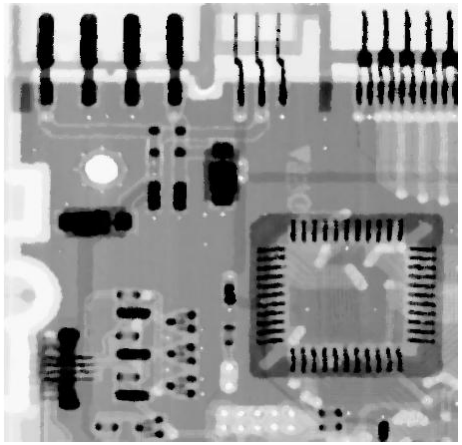
```
ordfilt2(u, (m*n+1)/2, ones(m,n))
```
**Vraća središnji element (medijan) iz pravougaone maske MxN.**

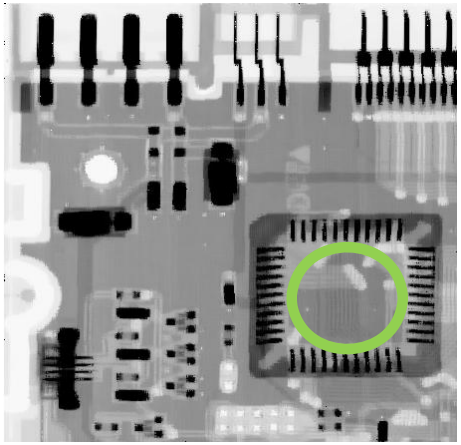
Za svaku poziciju maske sortira sve piksele koji pripadaju masci u neopadajući niz i prosledi na izlaz piksel koji ima odgovarajući redni broj u tako sortiranom nizu.

# Nelinearno filtriranje

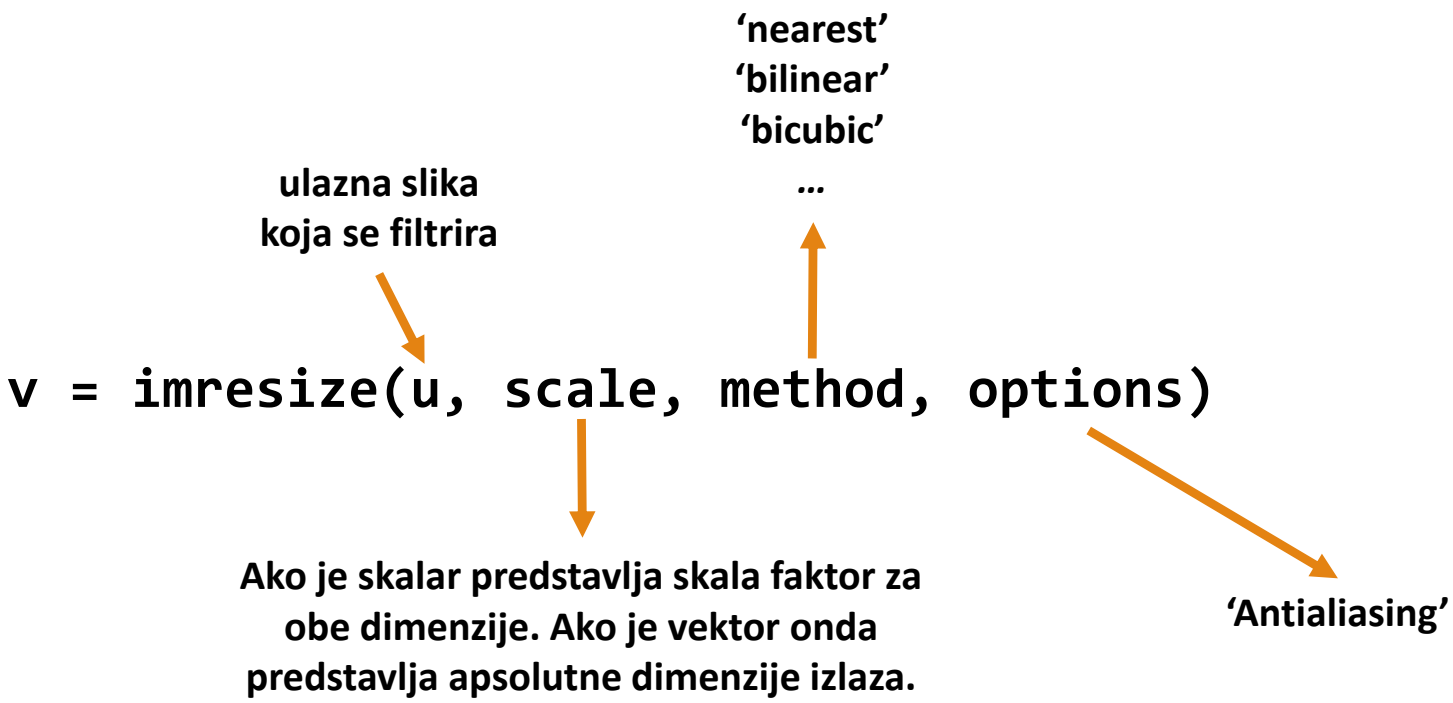
```
I = imread('ckt_board.tif');

figure; imshow(I);
set(gcf, 'Name', 'Zasumljena ulazna slika');

J = medfilt2(I, [5 5], 'symmetric');

figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika');

mask = zeros(9); mask(:,5)=1; mask(5,:) = 1;
J = ordfilt2(I, 9, mask);

figure; imshow(J);
set(gcf, 'Name', 'Izlazna slika2');
```







# Skaliranje slike

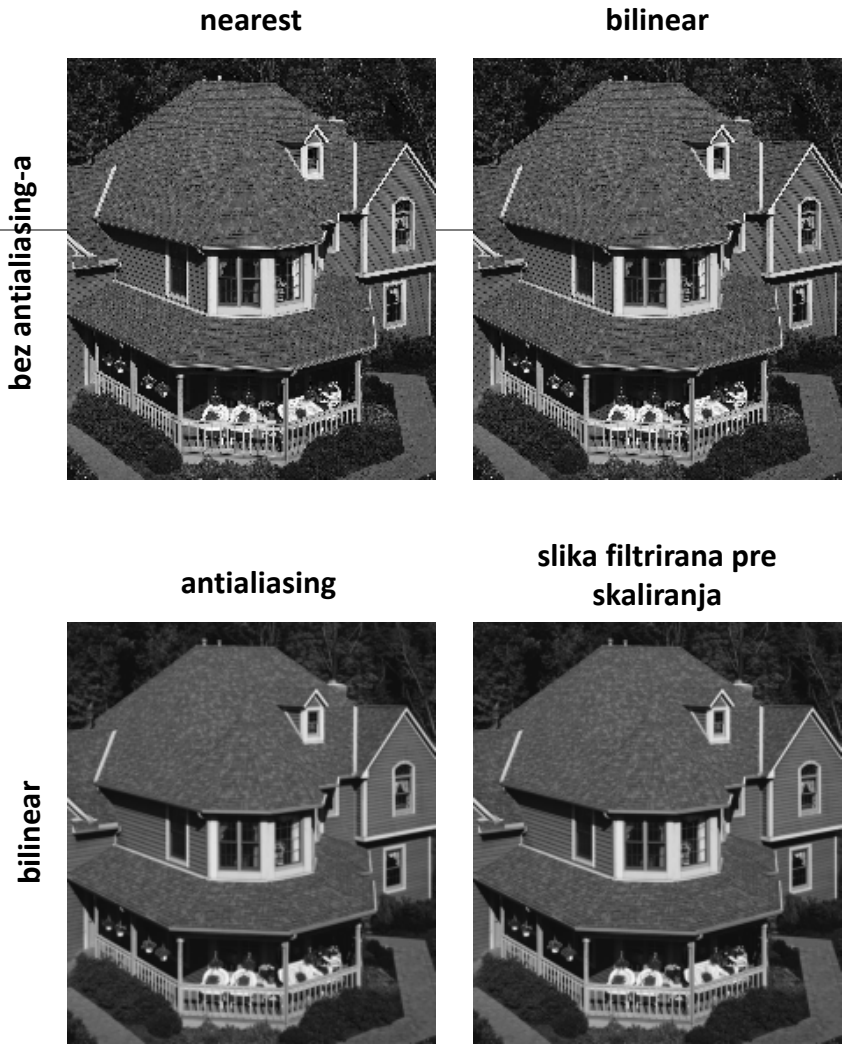


# Smanjivanje slike

```
f = imread('barbara.tif');

gn = imresize(f, 0.5, 'nearest', 'antialiasing', false);
gna = imresize(f, 0.5, 'nearest', 'antialiasing', true);
gbl = imresize(f, 0.5, 'bilinear', 'antialiasing', false);
gbla = imresize(f, 0.5, 'bilinear', 'antialiasing', true);

figure; imshow(gn);
figure; imshow(gna);
figure; imshow(gbl);
figure; imshow(gbla);
```

# Smanjivanje slike

```
I = imread('roof.jpg');

Jn = imresize(I, 0.2, 'nearest', 'Antialiasing', false);
Jb = imresize(I, 0.2, 'bilinear', 'Antialiasing', false);

figure; imshow(Jn);
figure; imshow(Jb);

Jba = imresize(I, 0.2, 'bilinear', 'Antialiasing', true);

If = imfilter(I, fspecial('gaussian', [9, 9], 2), 'replicate');
Jfb = imresize(If, 0.2, 'bilinear', 'Antialiasing', false);

figure; imshow(Jba);
figure; imshow(Jfb);
```



# Uvećanje slike (zumiranje)

```
f = imread('eye.png');

gn = imresize(f, 4, 'nearest');
gbl = imresize(f, 4, 'bilinear');
gbc = imresize(f, 4, 'bicubic');

figure; imshow(gn);
figure; imshow(gbl);
figure; imshow(gbc);
```

**ULAZ**

**NEAREST**

**BILINEAR**

**BICUBIC**

# Skaliranje slike – bez imresize

```
I = im2double(imread('eye.png'));

[M, N] = size(I);
scale = 5;
Ms = scale*M; Ns = scale*N; xs = 1:Ms; ys = 1:Ns;
x = (xs-1)/scale+1; y = (ys-1)/scale+1;
x0 = floor(x); y0 = floor(y); x1 = x0 + 1; y1 = y0 + 1;

Ip = padarray(I, [1, 1], 'post', 'replicate');

y_mat = repmat(y, Ms, 1);
y0_mat = repmat(y0, Ms, 1); y1_mat = repmat(y1, Ms, 1);

Jh0 = (y1_mat - y_mat).*Ip(x0, y0) + ...
(y_mat - y0_mat).*Ip(x0, y1);
Jh1 = (y1_mat - y_mat).*Ip(x1, y0) + ...
(y_mat - y0_mat).*Ip(x1, y1);

x_mat = repmat(x', 1, Ns);
x0_mat = repmat(x0', 1, Ns); x1_mat = repmat(x1', 1, Ns);

J = (x1_mat - x_mat).*Jh0 + (x_mat - x0_mat).*Jh1;

figure; imshow(J);
```

**ULAZ**

**5x BILINEAR**

# OPERACIJE NAD SLIKOM U BOJI

# Rad sa slikom u boji

1) Transformacija svakog od kanala boje nezavisno? Šta se dešava sa bojama?

2) Ista transformacija nad sva tri kanala? Šta ako je funkcija nelinearna?

3) Prelazak u neki od sistema HSV, YCbCr, Lab, rad na komponenti koja predstavlja komponentu osvetljaja i zatim vraćanje u RGB.

# Kompresija kontrasta slike u boji

```
I = double(imread('venice.jpg'))./255;

figure; imshow(I);
set(gcf, 'Name', 'Ulazna slika');

J = I.^0.6;
figure; imshow(J);
set(gcf, 'Name', 'Slika obradjena u RGB');

Iycbcr = rgb2ycbcr(I);
Iycbcr(:,:,1) = ((Iycbcr(:,:,1) - 16/255)*255/224).^0.6;
Iycbcr(:,:,1) = Iycbcr(:,:,1).*224/255 + 16/255;
J = ycbcr2rgb(Iycbcr); figure; imshow(J);
set(gcf, 'Name', 'Slika obradjena u YCbCr');

Ihsv = rgb2hsv(I);
Ihsv(:,:,3) = Ihsv(:,:,3).^0.6;
J = hsv2rgb(Ihsv); figure; imshow(J);
set(gcf, 'Name', 'Slika obradjena u HSV');

Ilab = applycform(I, makecform('srgb2lab'));
Ilab(:,:,1) = 100.*((Ilab(:,:,1)/100).^0.6);
J = applycform(Ilab, makecform('lab2srgb'));
figure; imshow(J);
set(gcf, 'Name', 'Slika obradjena u Lab');
```

ULAZ

RGB

HSV

YCbCr

LAB

# Filtriranje slike u boji

```
w = fspecial('gaussian', 101, 25);

J = imfilter(I, w, 'replicate');
figure; imshow(J);
set(gcf, 'Name', 'Slika filtrirana u RGB');

Ihsv = rgb2hsv(I);
Ihsv(:,:,3) = imfilter(Ihsv(:,:,3), w, 'replicate');
J = hsv2rgb(Ihsv);
figure; imshow(J);
set(gcf, 'Name', 'Slika filtrirana u HSV');

Iycbcr = rgb2ycbcr(I);
Iycbcr(:,:,1) = (Iycbcr(:,:,1) - 16/255)*255/224;
Iycbcr(:,:,1) = imfilter(Iycbcr(:,:,1), w, 'replicate');
Iycbcr(:,:,1) = Iycbcr(:,:,1).*224/255 + 16/255;
J = ycbcr2rgb(Iycbcr);
figure; imshow(J);
set(gcf, 'Name', 'Slika filtrirana u YCbCr');
```