

ROZ II – cv. 01

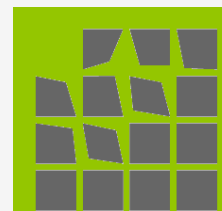
Dekonvoluce

KM - FJFI - ČVUT

ZS

ÚTIA - ZOI

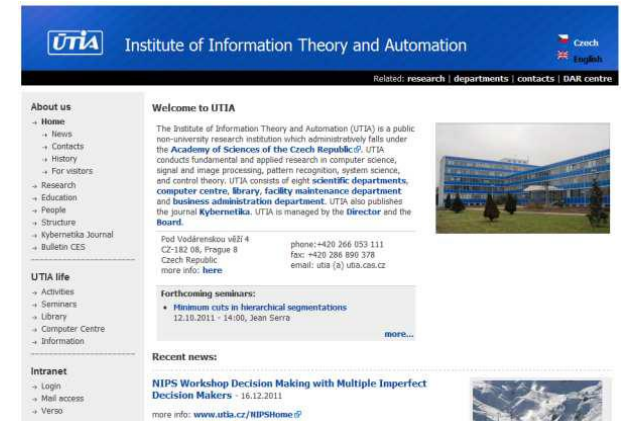
zoi.utia.cas.cz



Kontakty

- Ústav teorie informace a automatizace AV ČR, v.v.i.

- <http://www.utia.cas.cz>



- Zpracování obrazové informace

- <http://zoi.utia.cas.cz>

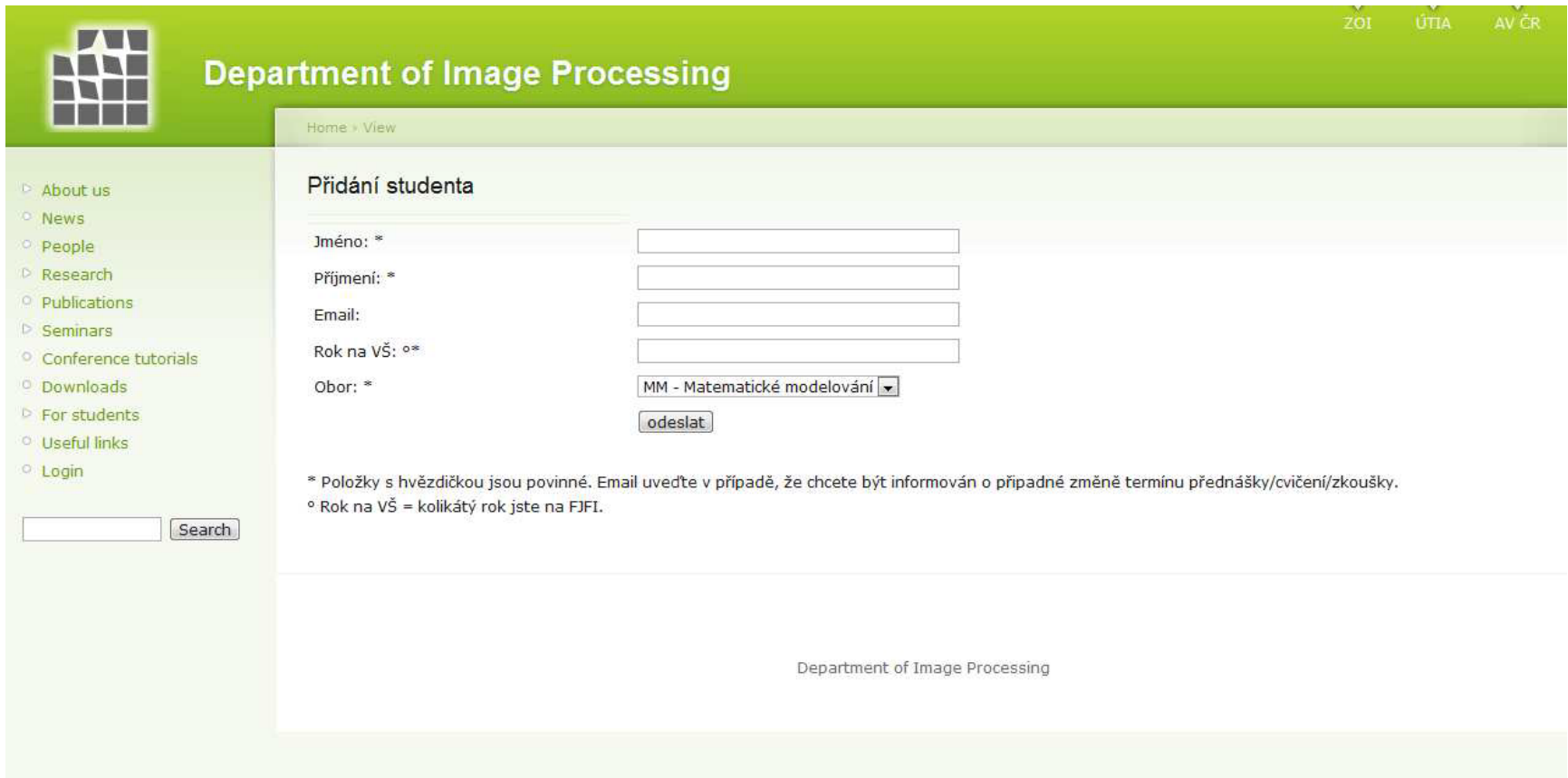


Kontakty

- Adam Novozámský: novozamsky@utia.cas.cz
- Jitka Kostková: kostkova@utia.cas.cz
 - UTIA místnost 23

Organizace cvik

- teorie + řešení úkolů
- docházka:
 - <http://zoi.utia.cas.cz> >> For students >> Lecture courses (CZ) >> ROZ2 >> Materiály >> Přidat studenta

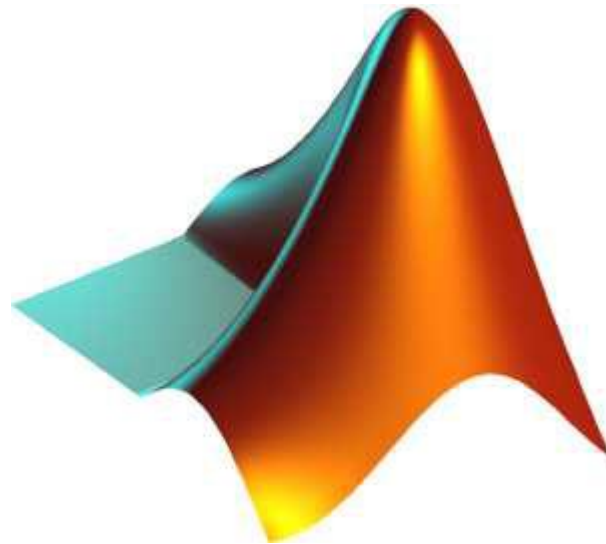


The screenshot shows the website of the Department of Image Processing. The header is green with the department name and navigation links (ZOI, ÚTIA, AV ČR). A left sidebar contains a menu with links like 'About us', 'News', 'People', 'Research', 'Publications', 'Seminars', 'Conference tutorials', 'Downloads', 'For students', 'Useful links', and 'Login'. The main content area is titled 'Přidání studenta' and contains a form with fields for 'Jméno: *', 'Příjmení: *', 'Email:', 'Rok na VŠ: °*', and 'Obor: *'. The 'Obor' dropdown is set to 'MM - Matematické modelování'. A 'odeslat' button is at the bottom of the form. Below the form, there are two footnotes: '* Položky s hvězdičkou jsou povinné. Email uveďte v případě, že chcete být informován o případné změně termínu přednášky/cvičení/zkoušky.' and '° Rok na VŠ = kolikátý rok jste na FJFI.' The footer of the page says 'Department of Image Processing'.

Department of Image Processing

Matlab

- Na UTIA není možné používat serverovou licenci z ČVUT
- Stáhnout balík souborů na cvičení:
 - <http://zoi.utia.cas.cz/ROZ2/studijni-materialy>



Maska Gaussiánu

- naprogramujte generovaní masky symetrického 2D Gaussiánu

```
function M = gauss2(S, N)
% M = gauss2 (S [,N]) - vraci normovanou
% symetrickou 2D gaussovku o smerodatne
odchylce Sigma
% N je velikost matice, defaultne 3*S+1
```

Maska Gaussiánu

- D-dimenzionální Gaussova funkce:

$$f(x, y) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(x-\mu)^t \Sigma^{-1} (x-\mu)\right]}$$

- d počet dimenzí
- μ vektor střední hodnoty
- $(\cdot)^t$ transpozice vektoru
- Σ diagonální kovarianční matice $\Sigma_{ii} = \sigma_i^2$
- $|\cdot|$ determinant

Maska Gaussiánu – 2D

- $|\Sigma|$

- $\det \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} = \sigma_1^2 \sigma_2^2 = \{\sigma_1 \equiv \sigma_2\} = \sigma_1^4$

- $(x - \mu)^t \Sigma^{-1} (x - \mu)$

- $(x_1 - \mu_1 \quad x_2 - \mu_2) \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} =$

- $= (x_1 - \mu_1 \quad x_2 - \mu_2) \begin{pmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} =$

- $= \begin{pmatrix} \frac{x_1 - \mu_1}{\sigma_1^2} & \frac{x_2 - \mu_2}{\sigma_2^2} \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} = \frac{(x_1 - \mu_1)^2}{\sigma_1^2} + \frac{(x_2 - \mu_2)^2}{\sigma_2^2} = \left\{ \begin{matrix} \sigma_1 \equiv \sigma_2 \\ \mu_1 \equiv \mu_2 = 0 \end{matrix} \right\} =$

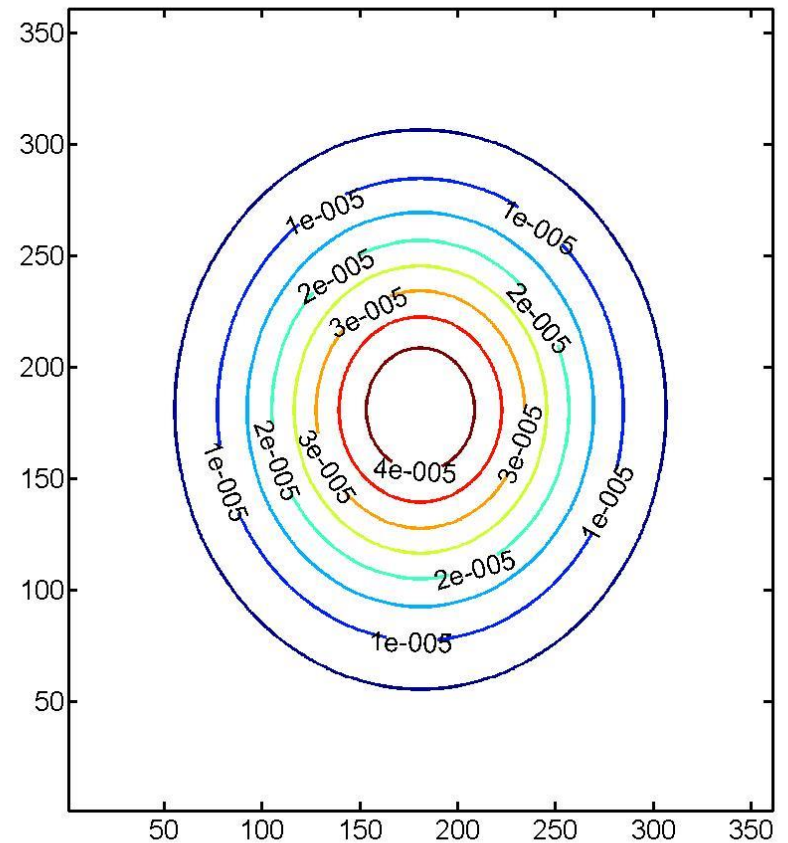
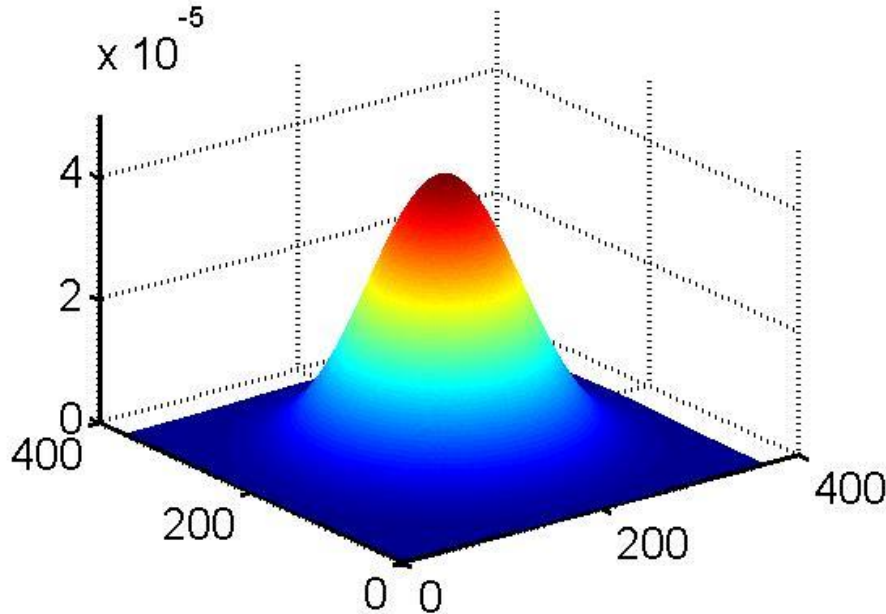
- $= \frac{x_1^2 + x_2^2}{\sigma_1^2}$

2D Gaussova funkce

- symetrická (izotropní)

$$\sigma_x \equiv \sigma_y = 0.6$$

$$\mu_x \equiv \mu_y = 0$$

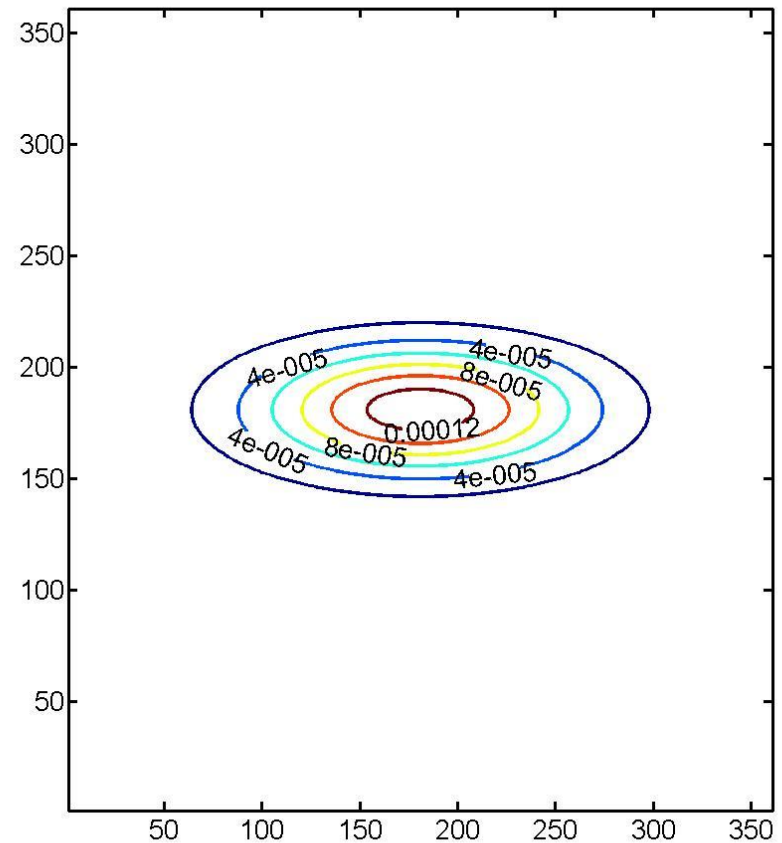
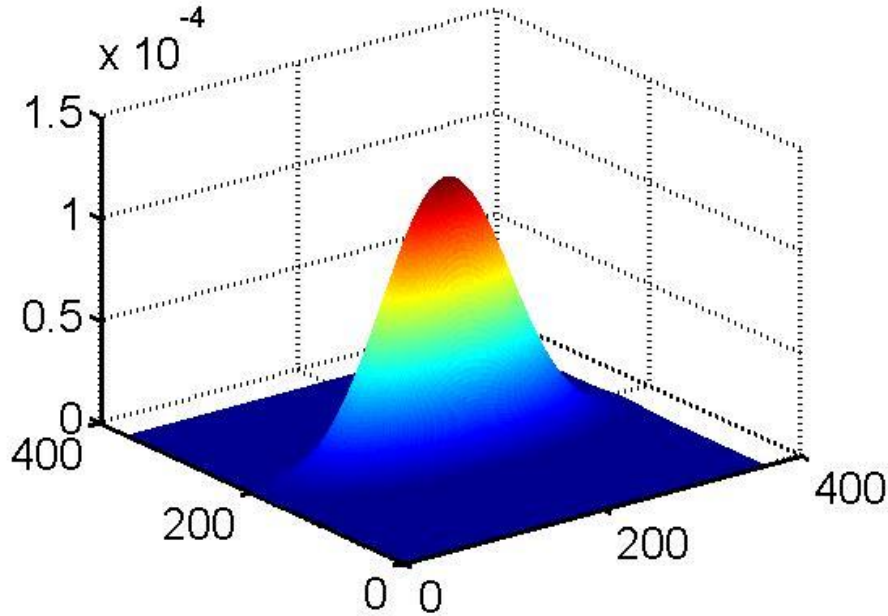


2D Gaussova funkce

- asymetrická (anizotropní)

$$0.6 = \sigma_x \neq \sigma_y = 0.2$$

$$\mu_x \equiv \mu_y = 0$$

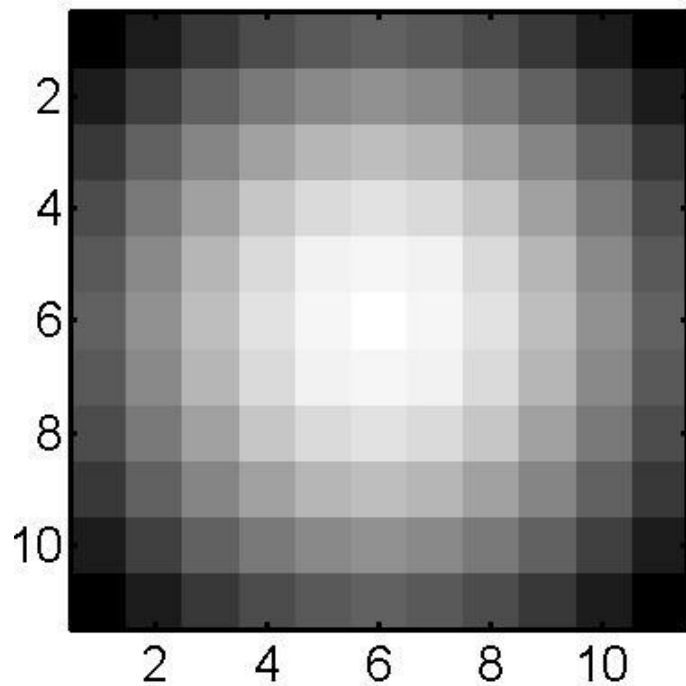


Maska Gaussiánu

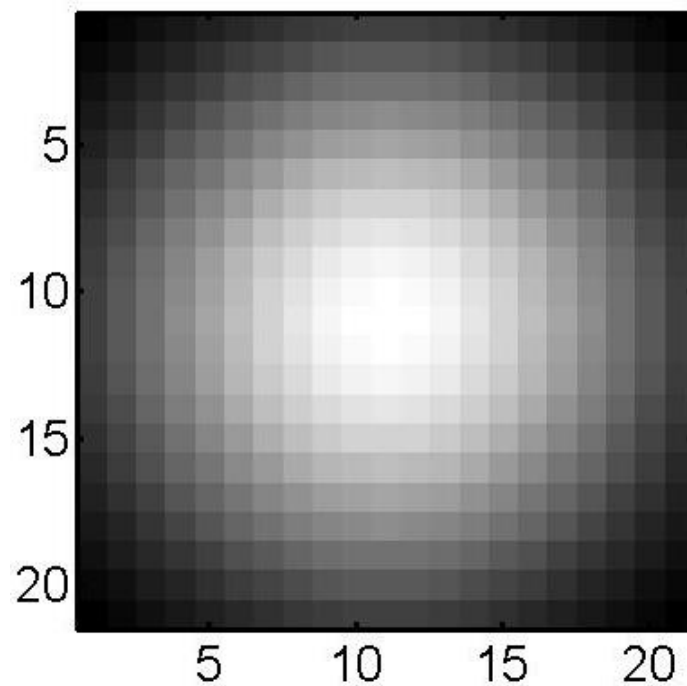
- naprogramujte generovaní masky anizotropního 2D Gaussiánu

```
function M = gauss2ani(S1, S2, N)
% M = gauss2 (S1, S2 [,N]) - vraci normovanou
% 2D gaussovku o smerodatnych odchylkach S1 a
% S2
% N je velikost matice, defaultne
% 3*max(S1*S2)+1
```

Maska Gaussiánu

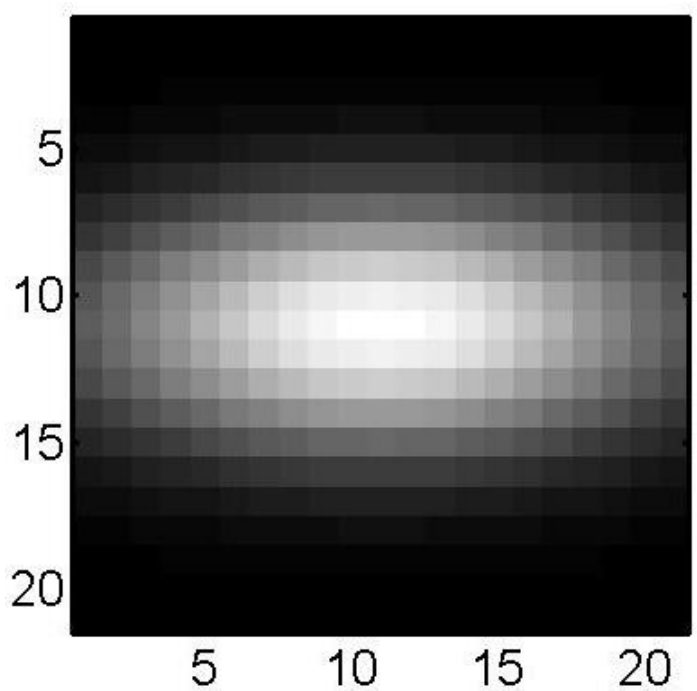


`gauss2 (5, 11)`

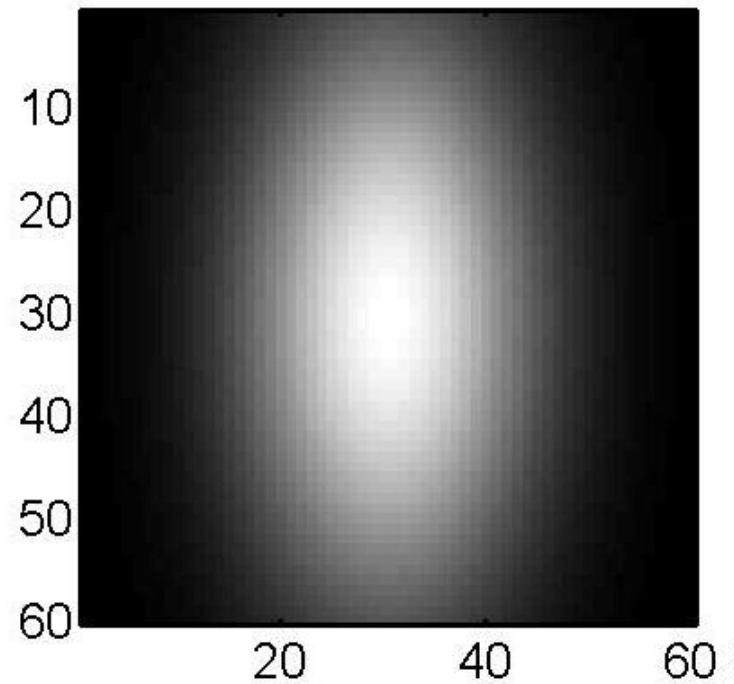


`gauss2 (7, 21)`

Maska Gaussiánu



`gauss2ani(7, 3, 20)`



`gauss2ani(10, 20, 60)`

Maska Gausiánu

```
function M = gauss2(S, N)
% M = gauss2 (S [,N]) - vraci normovanou 2D
% gaussovku o smerodatne odchylce S
% N je velikost matice, defaultne 3*S+1

if nargin >= 2
    R = (N-1) / 2;
else
    R = 3*S;
end

[X, Y] = meshgrid( -R:R);
M = (1/(2*pi*S^2))*exp( -(X.^2 + Y.^2) / (2 * S.^2) );
M = M / sum(M(:));
end
```

Maska Gausiánu

```
function M = gauss2ani(S1,S2, N)
% M = gauss2 (S1, S2 [,N]) - vraci normovanou
% 2D gaussovku o smerodatnych odchylkach S1 a S2
% N je velikost matice, defaultne 3*max(S1*S2)+1

if nargin >= 3
    R = (N-1) / 2;
else
    R = 3*max(S1,S2);
end

[X, Y] = meshgrid( -R:R);
M=(1/(2*pi*S1*S2))*exp((-1/2)*((X.^2/S1.^2)+(Y.^2/S2.^2)));
M = M / sum(M(:));
end
```

Přidání bílého šumu

- Signal-to-noise ratio (**SNR**) - míra šumu v obraze

- $SNR = 10 \log (D(f)/D(n))$ [dB]
 - $D(f)$... rozptyl nezašuměného signálu
 - $D(n)$... rozptyl šumu
- Ve frekvenční oblasti je SNR definována takto:
 - $\frac{|F|^2}{|N|^2}(u, v)$
 - Pokud je signál nekorelovaný $\Rightarrow |F|^2 = \sigma_f^2$
 - Kdyby šum byl bílý $\Rightarrow |N|^2 = \sigma_n^2$

- Odvození σ_n :

$$\frac{SNR}{10} = \log \frac{\sigma_f^2}{\sigma_n^2} \rightarrow 10^{\frac{SNR}{10}} = \frac{\sigma_f^2}{\sigma_n^2} \rightarrow \sigma_n^2 = \frac{\sigma_f^2}{10^{\frac{SNR}{10}}}$$

$$\sigma_n = \sqrt{\frac{\sigma_f^2}{10^{\frac{SNR}{10}}}}$$

... nápověda funkcí: `var()` , `rand()`

SNR u lena.pgm



20 dB

10 dB

0 dB

Přidání bílého šumu

- naprogramujte funkci, která přidá do obrázku bílý šum

```
function R = bilySum(I, SNR)
    % R = bilySum(I, SNR) - prida bily sum o danem
    % SNR do obrazku I
```

Přidání bílého šumu

```
function R = bilySum(I, SNR)
% R = bilySum(I, SNR) - prida bily sum o danem
% SNR do obrazku I

MinI = min(I(:));
MaxI = max(I(:));
S = sqrt(var(I(:)) / (10^(SNR/10)));
R = I + S*randn(size(I));
R(R<MinI) = MinI;
R(R>MaxI) = MaxI;
end
```

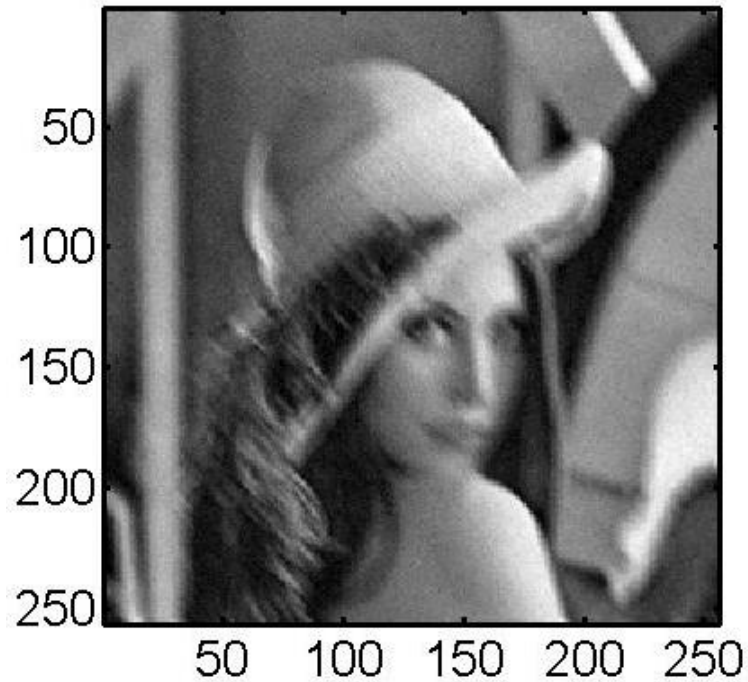
Poškození obrázku

- naprogramujte funkci, která poškodí obrázek – přidá bílý šum a poté obrázek rozmáže

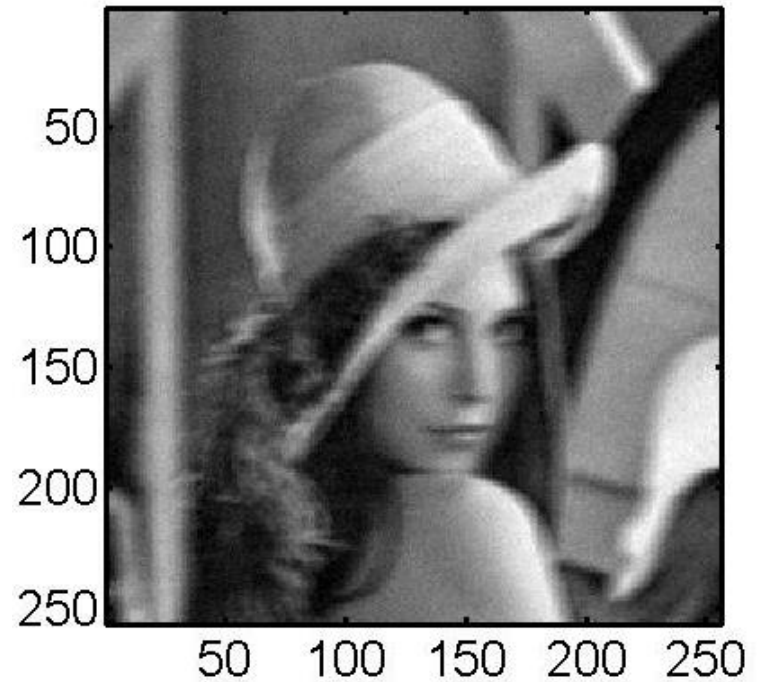
```
function R = damage(I, H, SNR)
```

```
% R = damage(I, H, SNR) - rozmaze obrazek I s konvolucni  
% maskou H a prida do nej bily sum o danem SNR
```

Maska Gaussiánu



damage (L, eye (7), 20)

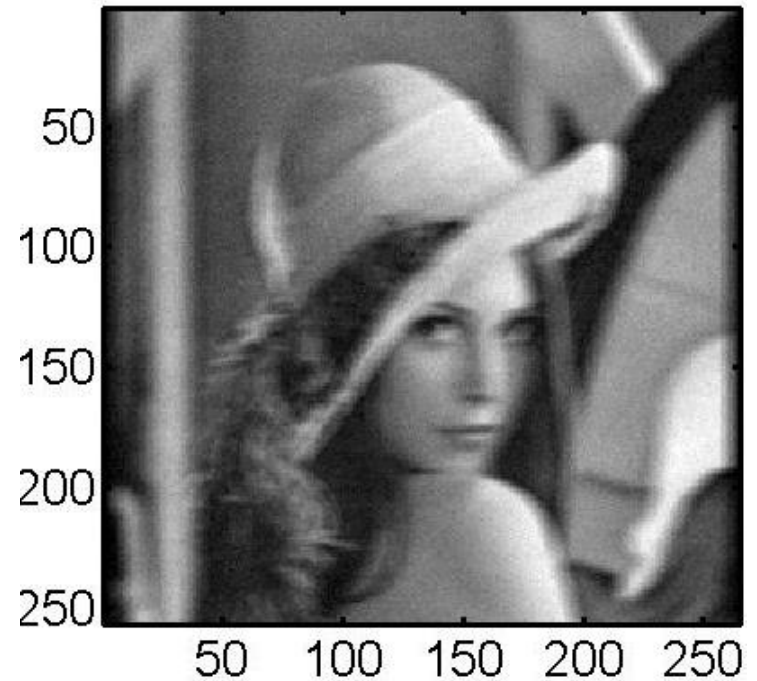


damage (L, ones (1,10), 20)

Maska Gaussiánu



`damage(L, eye(7), 20)`



`damage(L, ones(1,10), 20)`

Poškození obrázku

```
function R = damage(I, H, SNR)
% R = damage(I, H, SNR) - rozmaze obrazek I s konvolucni
% maskou H a prida do nej bily sum o danem SNR

R = conv2(I,H);
R = bilySum(R, SNR);
end
```

- „ubírá“ jas v okrajích >> tvoří rámeček
 - Jak ho odstranit?

Poškození obrázku

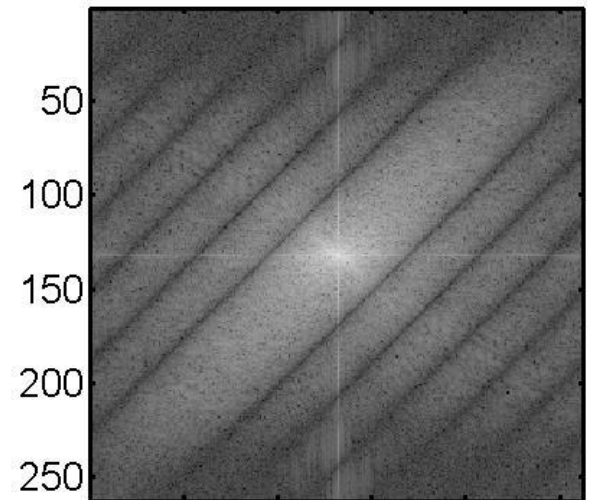
```
function R = damage2(I, H, SNR)
% R = damage(I, H, SNR) - rozmaze obrazek I s konvolucni
% maskou H a prida do nej bily sum o danem SNR

R = conv2(I, H, 'same') ./ conv2(ones(size(I)), H, 'same');
R = bilySum(R, SNR);
end
```


FT Poškozeného obrázku



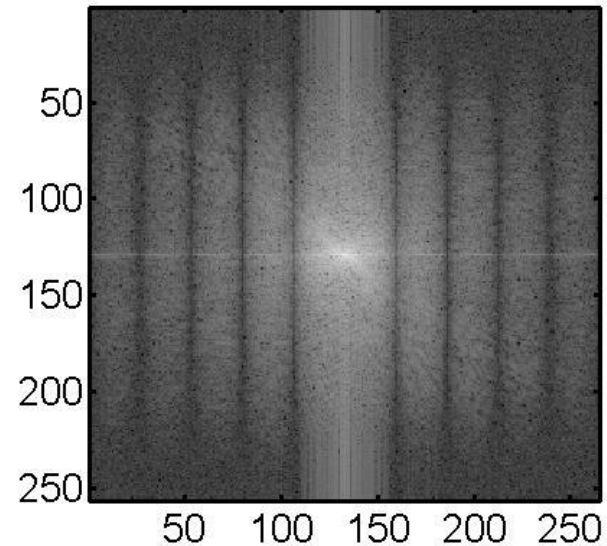
damage (L, eye (7), 50)



Proč tam jsou ty pruhy ?



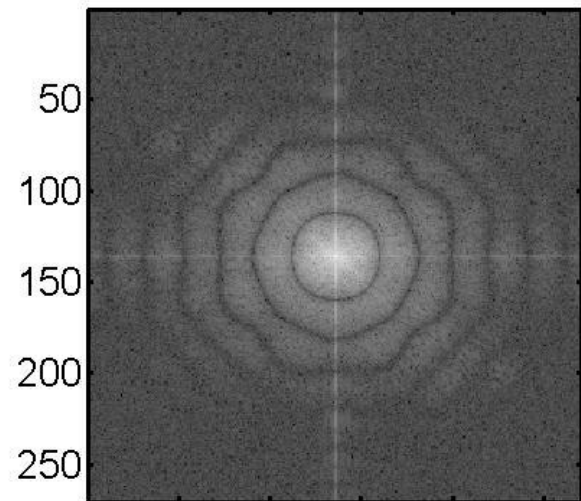
damage (L, ones (1,10), 50)



FT Poškozeného obrázku



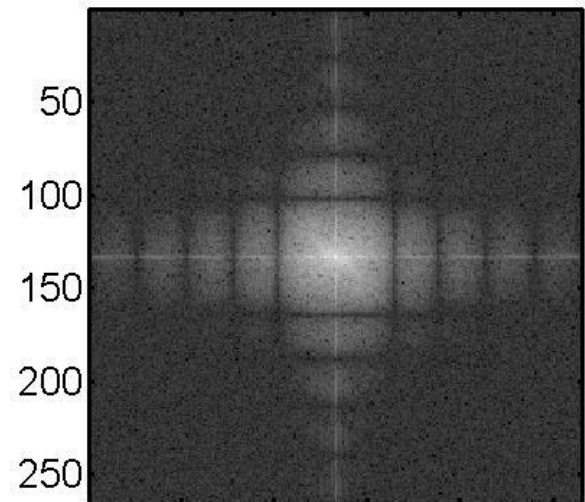
50 100 150 200 250
`damage(L, kruh(7,15), 50)`



Proč tam jsou ty vzory?

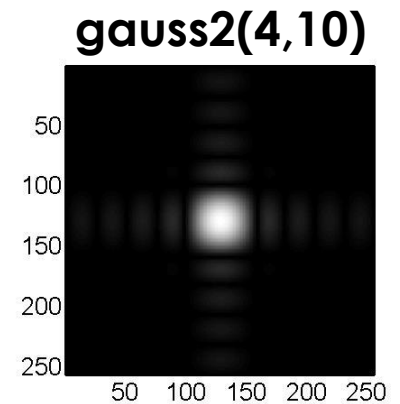
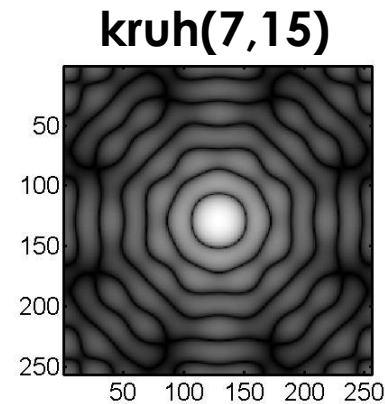
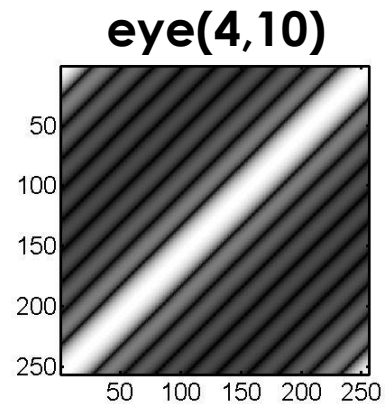
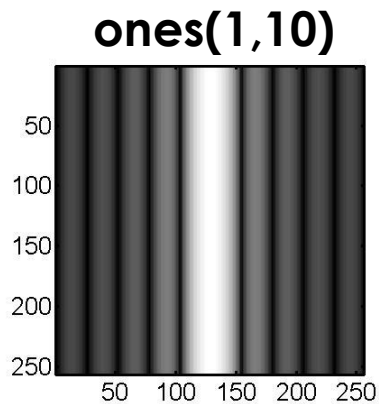


50 100 150 200 250
`damage(L, gauss2(4,10), 50)`



Proč jsou ve FFT ty periodické vzory u poškozených obrázků ?

- Z konvolučního teorému vyplývá, že konvoluce je násobení ve frekvenční oblasti.
- Konvoluční jádra mají ve FFT nulové body:



Poškození obrázku



horní řádek: `damage (L, gauss2 (3, 11) , SNR)`

dolní řádek: `damage (L, gauss2 (7, 21) , SNR)`

kde $\text{SNR} = 50, 20, 10, 5$

Inverzní filtr

- konvoluce:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$
$$*: L_1 \times L_1 \rightarrow L_1$$

- konvoluční teorém:

$$\mathcal{F}(f * g) = [\mathcal{F}(f)] \cdot [\mathcal{F}(g)] = F \cdot G$$

- odvození Inverzního filtru:

- $Z = U \cdot H$

- $U = \frac{Z}{H}$

- $u = \mathcal{F}^{-1}\left(\frac{Z}{H}\right)$

- naprogramujte Inverzní filtr:

```
function u = inverzniFiltr (z, h)
```

```
% u = inverzniFiltr (z, h) zaostří obrázek z
```

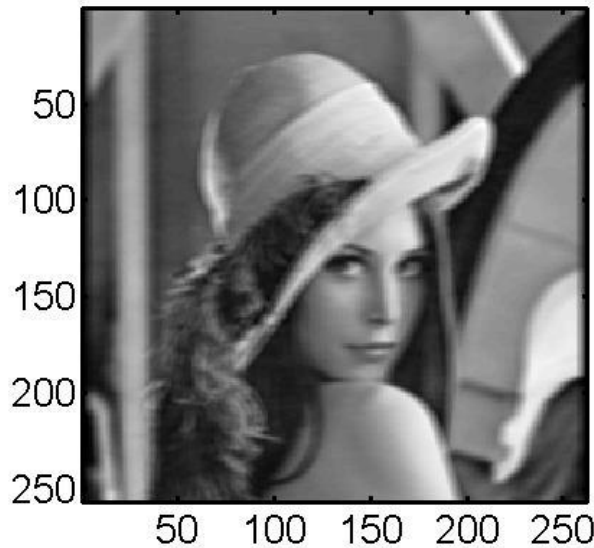
```
% rozmazaný maskou h pomocí inverzního filtru
```


Inverzní filtr

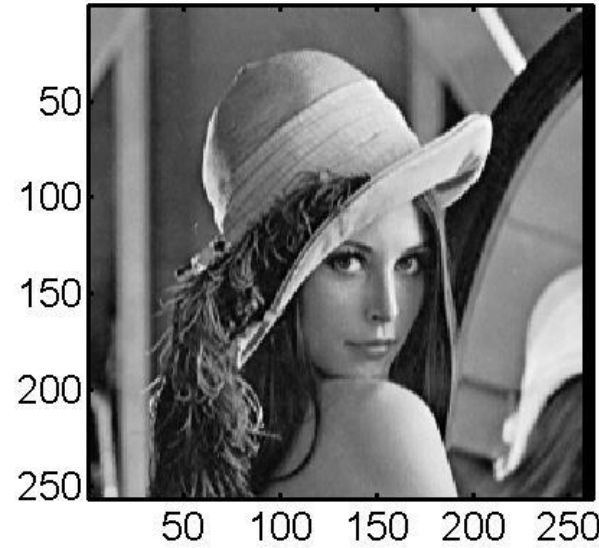
```
function u = inverzniFiltr (z, h)
    % u = inverzniFiltr (z, h) zaostří obrázek z
    % rozmazaný maskou h pomocí inverzního filtru

    Z = fft2 (z);
    H = fft2 (h, size(z,1), size(z,2));
    U = Z ./ H;
    u = abs (ifft2 (U) );
end
```

Inverzní filtr



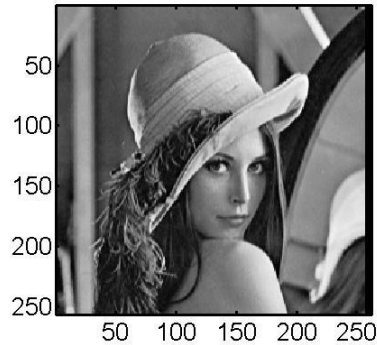
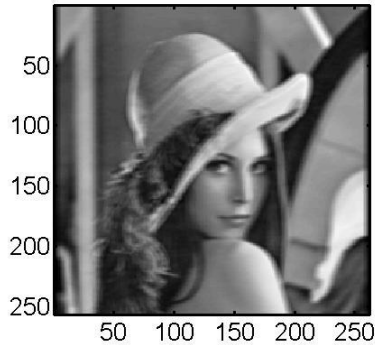
`C = conv2(L, ones(1,7))`



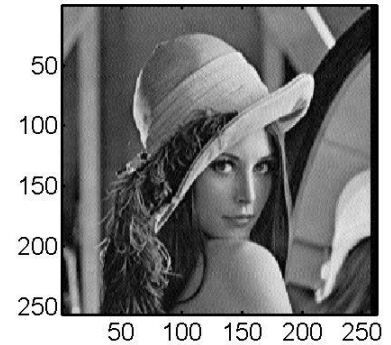
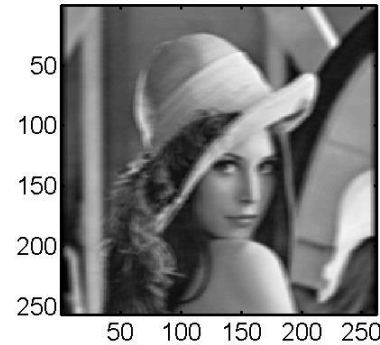
`inverzniFiltr(C,ones(1,7))`

Pro rozmazání (hlavně pohybem) je třeba použít velikost masky, která je nesoudělná s velikostí obrázku (např. 7). Jinak ve fourierce nafouklé na celý snímek vzniknou nuly, kterými se v inverzním filtru blbě dělí.

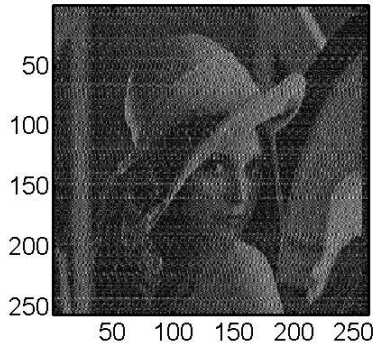
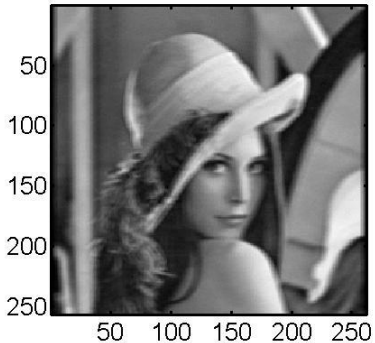
Inverzní filtr



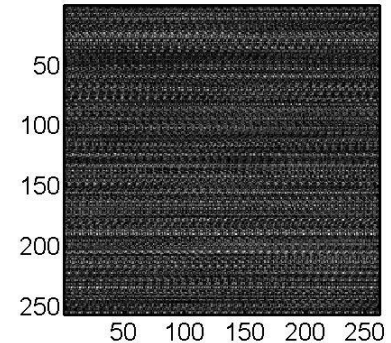
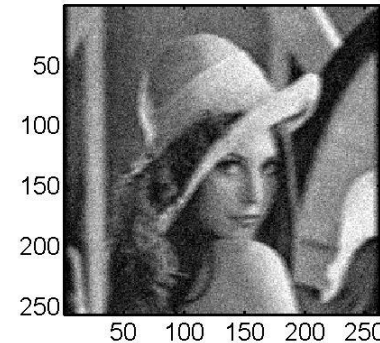
damage (L, ones (1 , 7) , 100)



damage (L, ones (1 , 7) , 50)



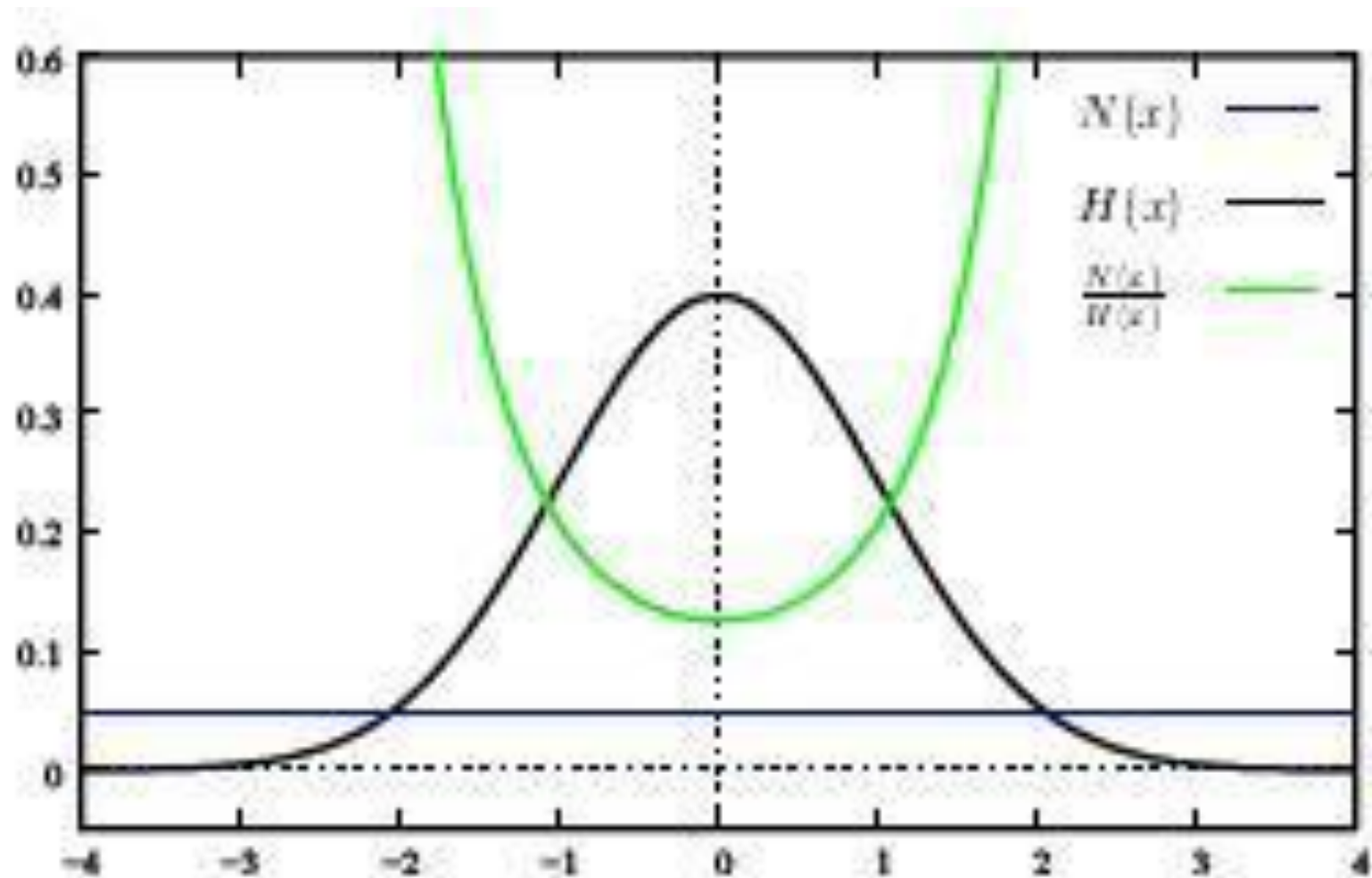
damage (L, ones (1 , 7) , 30)



damage (L, ones (1 , 7) , 10)

Přidání šumu má fatální vliv na Inverzní filtr !!
Nedá se v praxi moc použít !!

Inverzní filtr



Wienerův filtr

○ Definice:

- získaný odhad má mít minimální odchylku od originálu:
 - $E(\|f' - f\|^2) \rightarrow \min$... střední kvadratická chyba
 - E ...střední hodnota
 - f' ...odhad
 - f ...originál
- má to být lineární filtr, tedy to má být násobení ve frekvenční oblasti
 - $F' = G \cdot R$
 - G ... zašuměný obrázek
 - R ... transformační matice

Wienerův filtr

- byla odvozena tato transformační matice:

$$R(u, v) = \frac{1}{H(u, v)} \cdot \frac{|H(u, v)|^2}{|H(u, v)|^2 + S_n(u, v)/S_f(u, v)}$$

- $S_n(u, v)/S_f(u, v) \approx SNR^{-1}$
- $H \cdot \bar{H} = |H|^2$
 - Matlab funkce: `conj()`

$$R(u, v) = \frac{\overline{H(u, v)}}{|H(u, v)|^2 + 1/SNR}$$

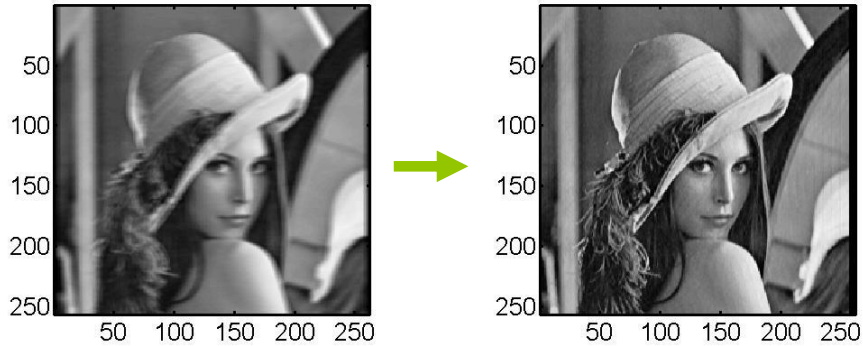
Wienerův filtr

- naprogramujte Wienerův filtr

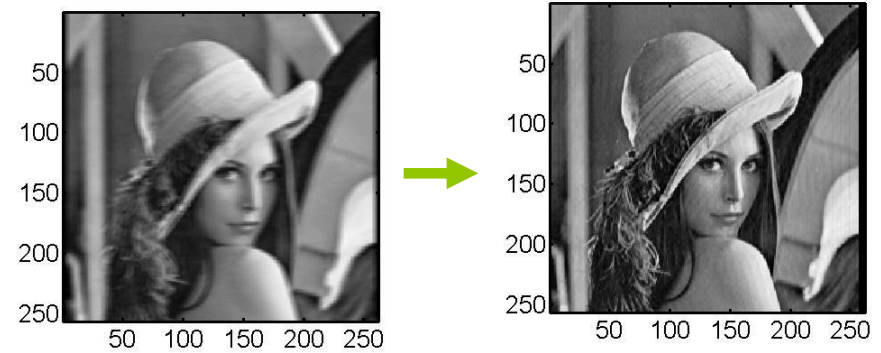
```
function u = wiener (z, h, SNR)
    % U = wiener (z, h, SNR) provede Wienerův
    % filtr na obrázek z, který byl poškozen
    % maskou h a zašuměn bílým šumem o daném SNR

    H = fft2( h, size(Img,1), size(Img,2) );
    W = conj(H) ./ (abs(H).^2 + 1/SNR);
    Z = fft2(z) .* W;
    u = abs(ifft2(Z));
end
```

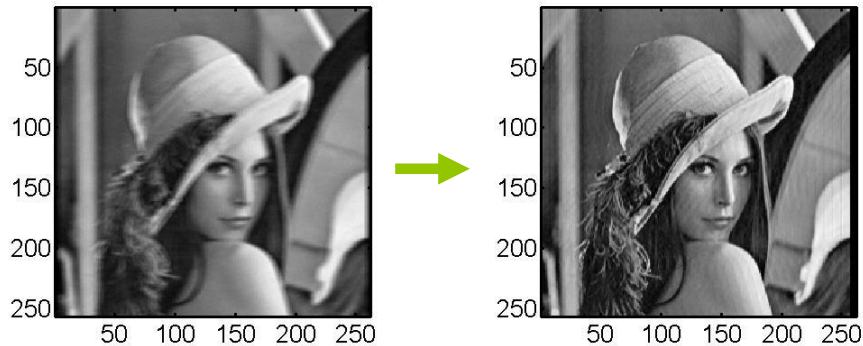
Wienerův filtr



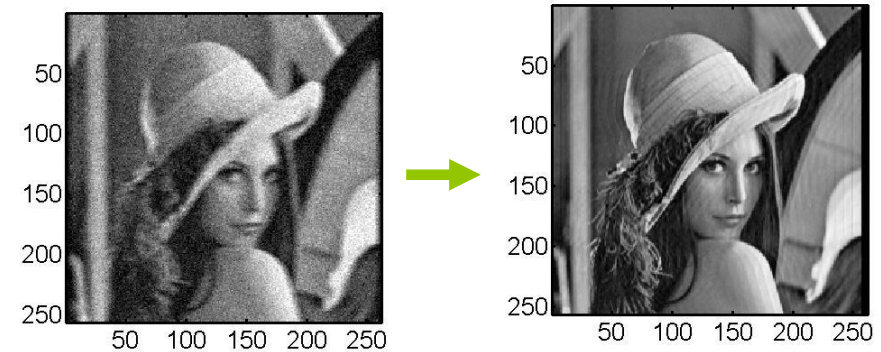
damage (L, ones (1 , 7) , 100)



damage (L, ones (1 , 7) , 50)



damage (L, ones (1 , 7) , 30)



damage (L, ones (1 , 7) , 10)

Praktické příklady

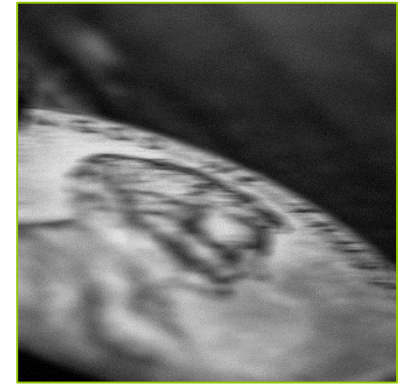
- Snažte se určit typ poškození u obrázků focus1.png, focus2.png a fokus3.png, tak aby na nich bylo možné přečíst text.



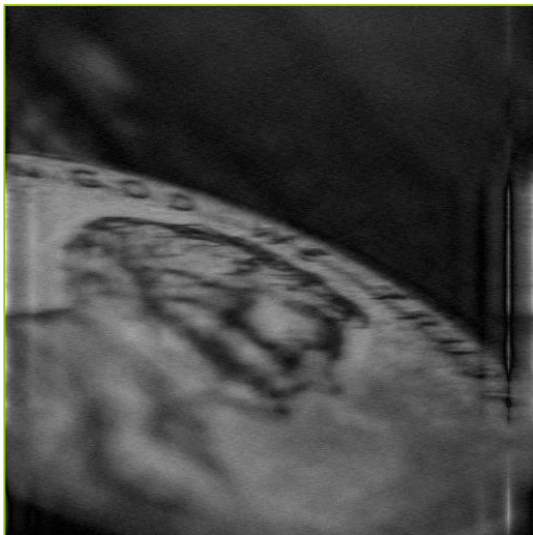
Praktické příklady



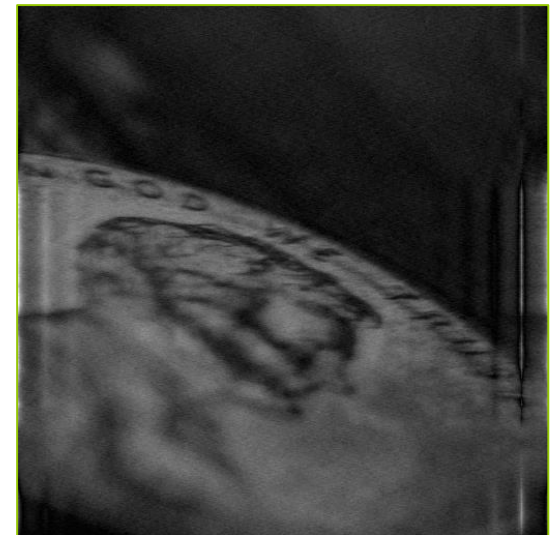
```
PSF = ones(1,26);  
PSFN = PSF / sum(PSF(:));  
SNR = 20;  
damage2(M, PSFN, SNR);
```



```
wiener(BN, PSFN, SNR);
```



```
wiener(BN, PSF, 1/SNR);
```



Praktické příklady



```
PSF = eye(13);  
PSFN = PSF / sum(PSF(:));  
SNR = 30;  
damage2(M, PSFN, SNR);
```



```
wiener(BN, PSFN, SNR);
```



```
wiener(BN, PSF, 1/SNR);
```



Praktické příklady



```
PSF = kruh(7,15);  
PSFN = PSF / sum(PSF(:));  
SNR = 40;  
damage2(M, PSFN, SNR);
```

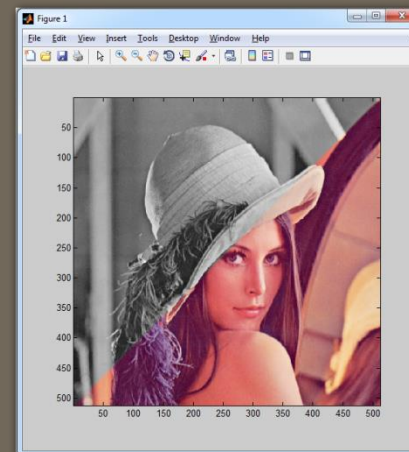


```
wiener(BN, PSFN, SNR);
```



```
wiener(BN, PSF, 1/SNR);
```





Děkuji za
pozornost
