Lab 5

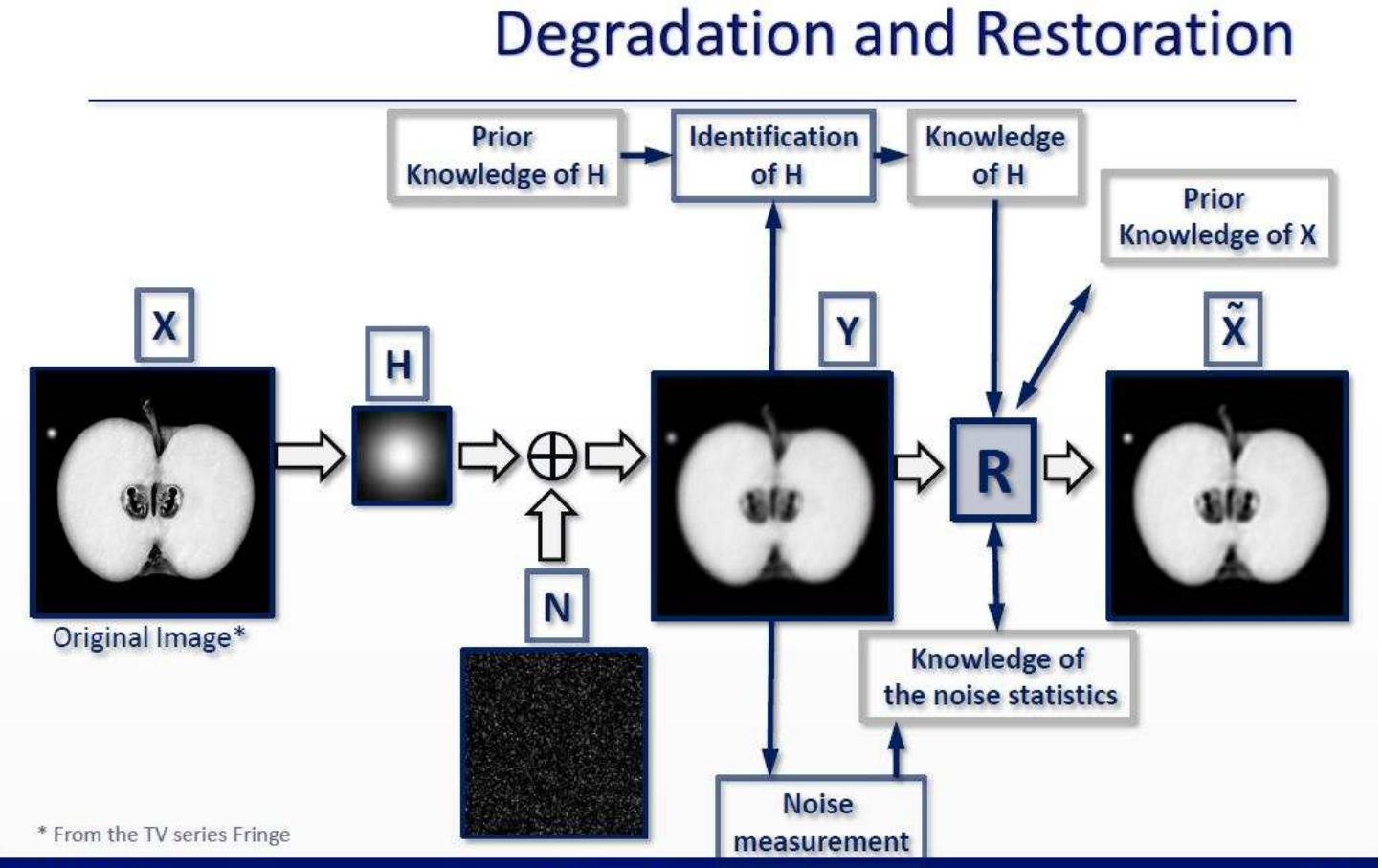
Basic Image Processing Algorithms
Fall 2016

Lab 5: Image recovery with Wiener-filter

- implement a Wiener-filter
- deadline (if you do not finish until the end of the lab): **23:59, 19/10/2016**;
  please send it to: bipa2016fall@gmail.com
  with the subject: LAB5
- students' most descriptive source-code comments from last year:

```
%wiener: Wiener Deconvolution Filter
%   This truly magnificient function can change everything
%   you perceive about what you see. In fact the amazing
%   performance can definetly redefine your definition of
%   of definitions. Has nothing to do with Wiener Schnitzels.
```

```
%   Mom get the camera
```



H: the system that introduces degradation

N: noise

Y: observed output

$y(n_1,n_2)=H[x(n_1,n_2)]+n(n_1,n_2)$

Wiener Filter

- Stochastic restoration approach:
  - Treat the image as a sample from a 2D random field.
  - The image is part of a class of samples (an ensemble), realizations of the same random field.

$$R(\omega_1,\omega_2)=\frac{H^*(\omega_1,\omega_2)\cdot P_{xx}(\omega_1,\omega_2)}{\left|H(\omega_1,\omega_2)\right|^2\cdot P_{xx}(\omega_1,\omega_2)+P_{NN}(\omega_1,\omega_2)}$$

  - Autocorrelation:

Expected value over many realizations of the 2D random field

$$R_{ff}(n_1,n_2,n_3,n_4)=E\left[f(n_1,n_2)f^*(n_3,n_4)\right]$$

  - Power-Spectrum:

Fourier transformation

$$P_{ff}(\omega_1,\omega_2)=\mathrm{F}\left\{R_{ff}(d_1,d_2)\right\}$$

we have the assumption that the input has the property of **Wide Sense Stationarity (WSS)** which generalized definition as follows:

$R_{ff}(n_1,n_2,n_3,n_4) =$

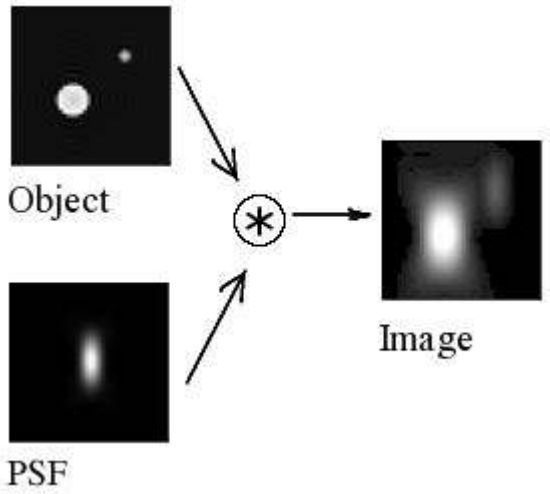
$R_{ff}(n_1\text{-}n_3,n_2\text{-}n_4) =$

$R_{ff}(d_1,d_2)$

Recap

Point spread function: describes the response of an imaging system to a point source or point object. A more general term for the PSF is a system's impulse response (the impulse response of a focused optical system). The degree of spreading (blurring) of the point object is a measure for the quality of an imaging system.

Optical transfer function: the Fourier transform of the PSF.



Recap

Estimation of **Noise-to-signal power ratio=**
variance_of_noise / variance_of_original_image

Autocorrelation (1D)**:** the cross-correlation of a signal with itself at different points in time.

$$R_{ff}(\tau) = \int_{-\infty}^{\infty} f(u)\overline{f}(u-\tau)\,\mathrm{d}u$$

Autocorrelation function (2D - in practice)**:**
the Fourier Transform of the image is multiplied by its complex conjugate, then the product is inverse Fourier Transformed back to normal domain.

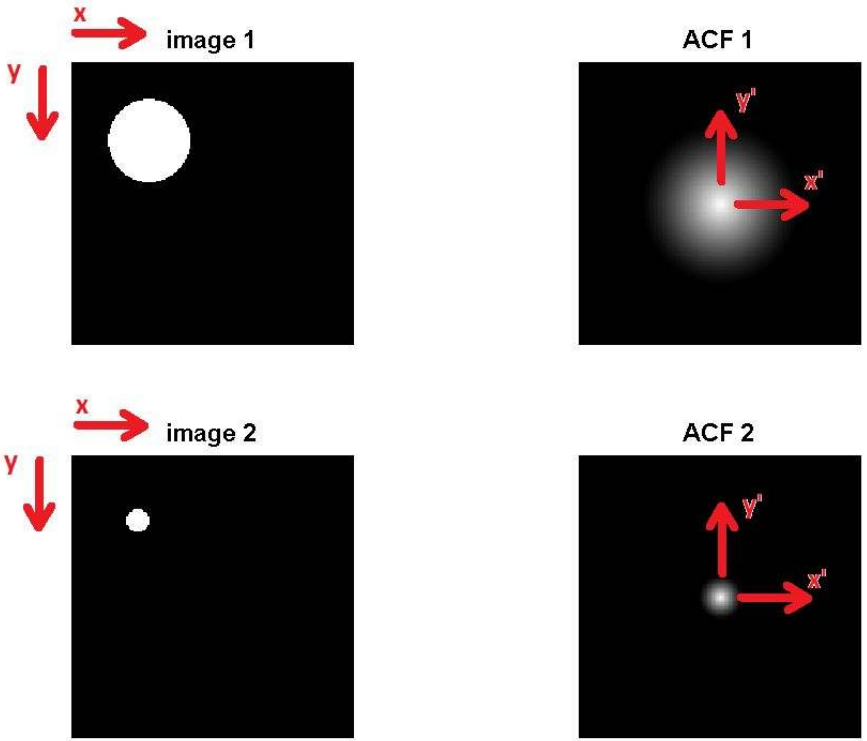
$F_R(f) = \mathbf{FFT}[X(t)]$ FT of the image

$S(f) = F_R(f)F_R^*(f)$ power spectrum, * means complex conj.

$R(\tau) = \mathbf{IFFT}[S(f)]$ autocorrelation

Recap

Autocorrelation function (ACF, 2D)



- Wiener filter:

$$R(\omega_1,\omega_2)=\frac{H^*(\omega_1,\omega_2)\cdot P_{xx}(\omega_1,\omega_2)}{\left|H(\omega_1,\omega_2)\right|^2\cdot P_{xx}(\omega_1,\omega_2)+P_{NN}(\omega_1,\omega_2)}=$$

$$=\frac{H^*(\omega_1,\omega_2)}{\left|H(\omega_1,\omega_2)\right|^2+\frac{P_{NN}(\omega_1,\omega_2)}{P_{xx}(\omega_1,\omega_2)}}=\frac{H^*(\omega_1,\omega_2)}{\left|H(\omega_1,\omega_2)\right|^2+\frac{\sigma_N^2}{P_{xx}(\omega_1,\omega_2)}}$$

Assuming white noise

H: the optical transfer function of the system, produced from psf
first realize it with the MATLAB built-in command `psf2otf`
H*: the complex conjugate of H itself (use the built-in `conj`)
P_{NN}: the noise power spectrum (FT of the noise's ACF, see slide #4)
P_{XX}: the signal power spectrum (FT of the signal's ACF)
the last term in the denominator (circled with red) can be replaced with the noise-to-signal power ratio

Exercise 1

Create a function that:

- can have 3 or 4 input parameters (use ***varargin***):
  - *if 3 input:* input_img_matrix, psf, noise_to_signal_power_ratio
  - *if 4 input:* input_img_matrix, psf, noise_autocorr, signal_autocorr
  - *output:* Wiener-filtered (restored) image,
- it should work with grayscale images,
- **output_img_matrix = | IFT(R · FT(input_img_matrix)) |** where
  - FT means Fourier Transform,
  - IFT means inverse Fourier Transform,
  - R stands for the Wiener-filter,
  - dot (·) in this case just a multiplication, since you are in frequency-domain,
  - |...| is just the absolute-value.

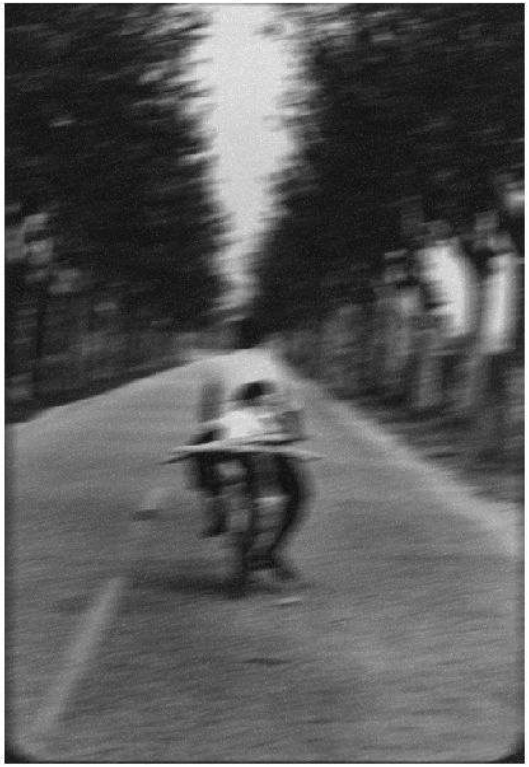
Exercise 2

Create a script that:

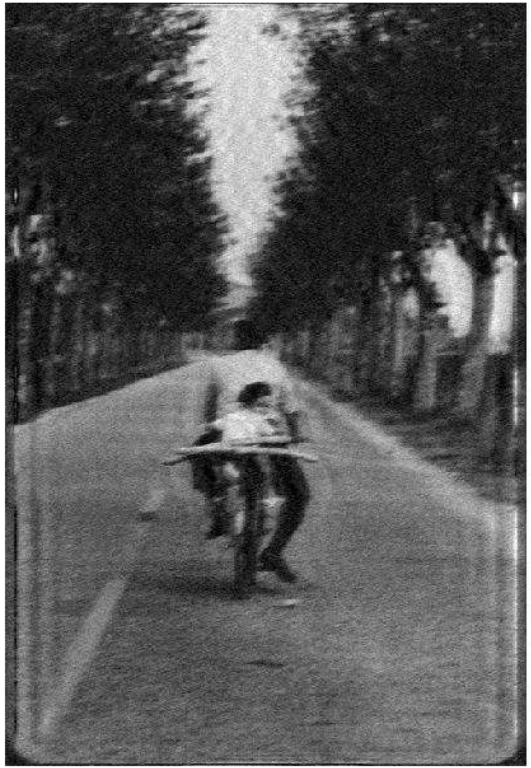
- reads in ElliottErwitt_Provence.jpg, converts it to grayscale (if necessary) and converts it to double (right now with `im2double` -- this will change the range of values as well to [0 1])
- blurs the image (motion blur with pixel shifts of 21 and rotation of degrees 11; use `fspecial` to create the psf of our 'capturing system', then use `imfilter`)
- create a noise layer separately, and add it to the blurred image (`imnoise`, Gaussian-type with zero mean and `0.001` variance)
- calls your Wiener filter function in two different cases:
  - **case1:** with **noise-to-signal power ratio** (which is the variance of noise divided by the variance of the input image; to compute the variance of the input please use `var`),
  - **case2:** with the **ACF of your noise-layer** (what you created separately) and with the **ACF of the input image** (in this case use `edgetaper` on the noisy-and-blurred image to make smoother the edge/frame-regions of your image; later we will reproduce this edgetaper function),
- pops up a figure with 3 subplots: original img, noisy img, reconstructed img.

original

noisy blurred

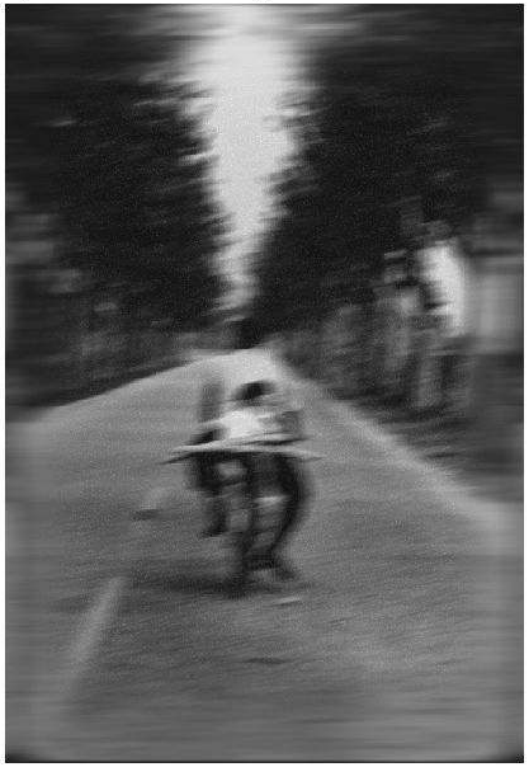


reconstructed with nsr



original

noisy blurred with edge-/frame-modification

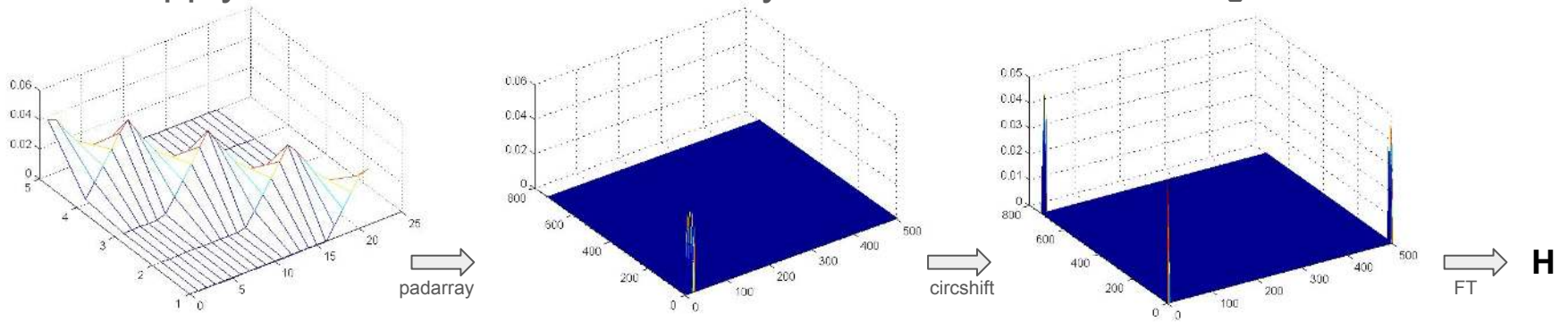


reconstructed with autocorrelations

How to replace `psf2otf` (optional)

instead of `H = psf2otf(psf, size(in_img))` please do the followings:

- resize your `psf` to the size of your input image---this does not mean rescaling, rather just extending your psf with zeros on the right and at the bottom (use `padarray` with '`post`' option)
- circularly shift your resized `psf` to have the center element of it at position (1,1) (use `circshift` with the half-sizes of your original psf, in negative direction both; the reason to do so: MATLAB FT will treat cell (1,1) being in the upper-left corner of your matrix, not in the center of it)
- apply Fourier Transformation on your resized and shifted `psf`



How to replace `edgetaper` (optional)

instead of

```
psf2 = fspecial('motion', 50, 0);
noisy_blurred_img2 = edgetaper(noisy_blurred_img, psf2);
```

please do the followings:

- create `psf2` as above :)
- create the optical transfer function (`otf`) of this `psf2` (as described in the previous slide)
- create a blurred version of your input image with this `otf`, the way how to do it:
  tmp_img = | IFT(FT(noisy_blurred_img) · otf) |
- create a weighting-matrix (now call it `wm`), with which you can create the weighted sum of your `noisy_blurred_img` and your `tmp_img`
- `noisy_blurred_img2 =`

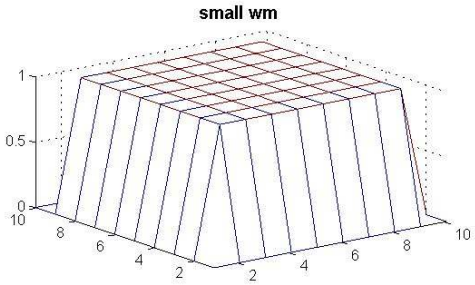
  `wm*noisy_blurred_img + (1-wm)*tmp_img`

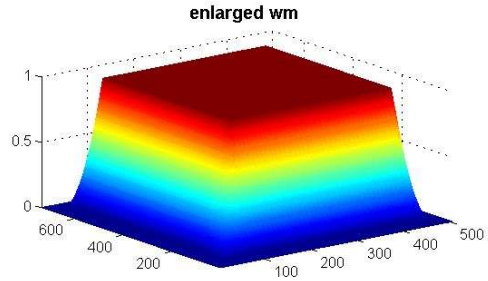
see next slide

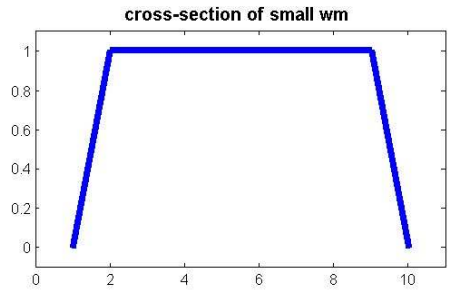
How to replace **`edgetaper`** (optional) - continued

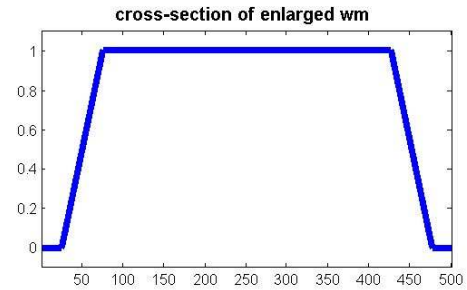
creating your weighting matrix (**`wm`**):

- create a small matrix (eg 10x10) containing ones,
- set to zero all of its boundary cells,
- resize it with **`imresize`** to the size of your image, use the '**`bilinear`**' option
- regularize the outliers :) namely: elements greater than 1 should be 1, elements smaller than 0 should be 0.









Further readings

- https://en.wikipedia.org/wiki/Point_spread_function
- https://en.wikipedia.org/wiki/Optical_transfer_function
- https://earth.unibas.ch/micro/workshops/warsaw/PDFs/digi05_print.pdf
- (http://www.ucl.ac.uk/~ucbpmbc/otf.html)