Assignment 1

Basic Image Processing Algorithms
Fall 2016

General informations

- during the semester there will be 5-6 programming assignments with different level of complexity - for different number of assignment-scores
- the total number of scores from all of the assignments will be between 130-170 points
    - you have to reach the level of 50 points to be able to participate to the exam (required minimum level)
    - you have to reach the level of 80 points as a necessary condition to Offered Final Grade
- the deadlines for the different assignments will be published separately

Assignment 1: Wallis operator OR anisotropic diffusion

- you can choose only one from the following two:
  - Wallis operator
  - anisotropic diffusion
- the number of scores to this assignment: 10 points
- deadline: **23:59, 19/10/2016**;
  please send it to: bipa2016fall@gmail.com
  with the subject: ASSIGNMENT1

Wallis Operator

- The Wallis operator can help to adjust local contrast:

$$y(n_1, n_2) = [x(n_1, n_2) - \bar{x}(n_1, n_2)] \frac{A_{\max}\sigma_d}{A_{\max}\sigma_l(n_1, n_2) + \sigma_d} + [p\bar{x}_d + (1-p)\bar{x}(n_1, n_2)]$$

  - where σ_l the local contrast: $\sigma_l(n_1, n_2) = \frac{1}{|N|}\sqrt{\sum_{(n_1,n_2)\in N}\sum (x(n_1, n_2) - \bar{x}(n_1, n_2))^2}$
  - $\bar{x}$ is the local average: $\bar{x}(n_1, n_2) = \frac{1}{|N|}\sum_{(n_1,n_2)\in N}\sum x(n_1, n_2)$
  - σ_d is the desired local contrast, $\bar{x}_d$ is the desired mean value of all pixels, p is a weighting factor of the mean compensation, while A_{max} is maximizing the local contrast modification.

Wallis Operator

- We can describe the image the following way:

$$x(n_1, n_2) = \underbrace{\left[x(n_1, n_2) - \bar{x}(n_1, n_2)\right]}_{(1)} + \underbrace{\bar{x}(n_1, n_2)}_{(2)}$$

where (2) is the local mean and (1) is the deviation from the local mean.

- With the transformation we want to „push" the local mean and standard deviation to a predefined desired value:

$$y(n_1, n_2) = \underbrace{\left[x(n_1, n_2) - \bar{x}(n_1, n_2)\right] \frac{\sigma_d}{\sigma_l(n_1, n_2)}}_{(1)} + \underbrace{\left[p\bar{x}_d + (1-p)\bar{x}(n_1, n_2)\right]}_{(2)}$$

- We are almost there, but if the local contrast is too low, the weighting in (1) may get too high, this is why we maximize it with A_{max}:

$$\frac{\sigma_d}{\sigma_l(n_1, n_2)} \Rightarrow \frac{A_{\max}\sigma_d}{A_{\max}\sigma_l(n_1, n_2) + \sigma_d}$$

Wallis operator

Create a function that:

- realizes the Wallis operator,
- parameters of this function should be as follows:
  - *input1:* image matrix,
  - *input2:* desired mean value $(\bar{x}_d)$
  - *input3:* desired variance (σ_d),
  - *input4:* A_{max},
  - *input5:* ratio (variable p on the slide),
  - *input6:* size of the local neighborhood (eg 3 for 3x3 window),
  - *output:* Wallis filtered image,
- it should work with grayscale images (check if this holds).

Wallis operator - continued

Create a script that:

- reads in this image: AndreKertesz_Paris_ManOnBicycle_part.jpg,
- converts it to grayscale,
- creates a blurred version of the input image (first create a motion-kernel with **`fspecial`**, eg 5 pixels shift, 10 degrees rotation; then modify your input image with **`imfilter`**),
- calls your Wallis filter on the blurred image (parameters eg:
  $\bar{x}_d = 128, \quad \sigma_d = 100, \quad A_{max} = 4, \quad p = 0.2,$ *neighborhood size: 9*;
  but please try other parameter-values as well),
- displays the input image, the blurred and the filtered images as well, see next slide:

Wallis operator - continued

original input

blurred image

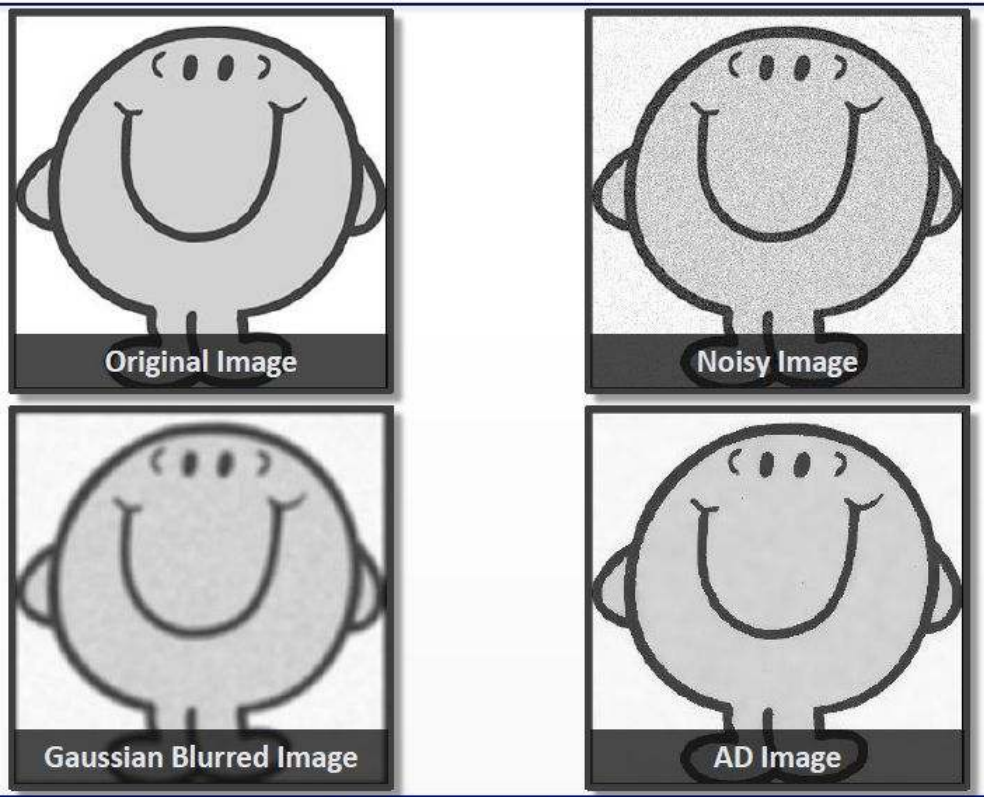
Wallis filtered image

Anisotropic Diffusion

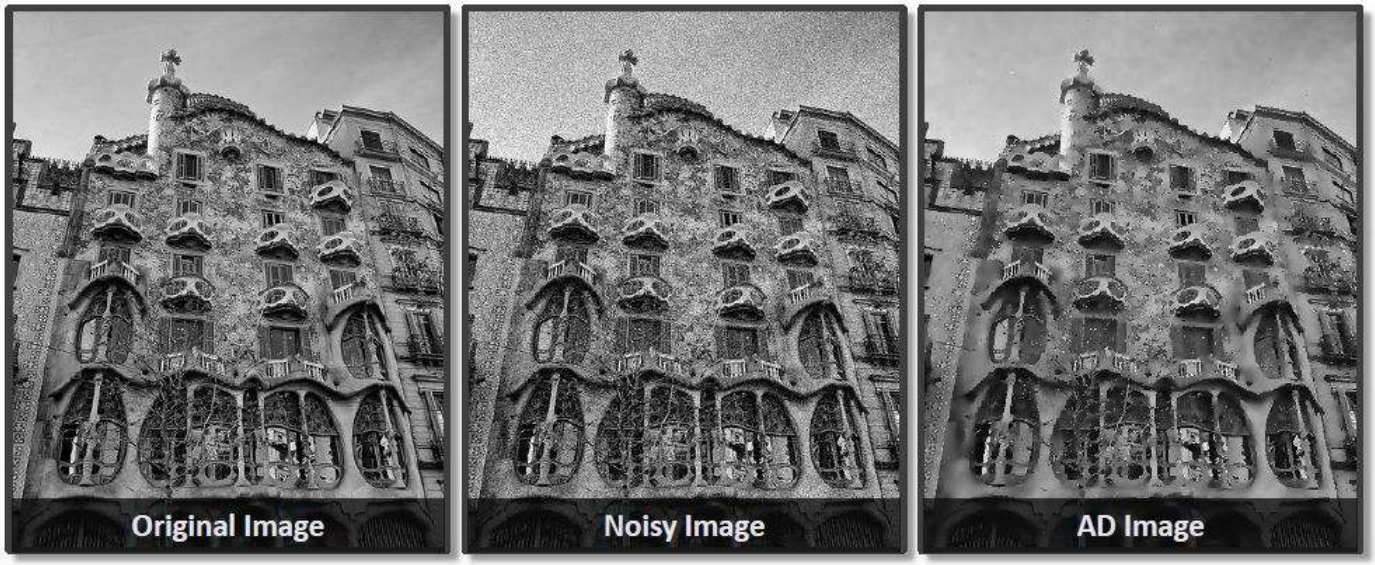
- The anisotropic diffusion is a technique aiming at reducing image noise without blurring significant parts of the image content.
- It was first proposed by D. Gábor in 1965 and later by Perona and Malik around 1990.
- ***Non-linear*** and ***space-variant*** transformation.
- The main idea is that the effect of blurring in each direction is inversely proportional to the gradient value in that direction:
  - allows diffusion along the edges or in edge-free territories, but penalizes diffusion orthogonal to the edge direction.
- AD is an iterative process

P. Perona, J Malik (July 1990). "Scale-space and edge detection using anisotropic diffusion". IEEE Tr. PAMI, 12 (7): 629–639.
D. Gabor, "Information theory in electron microscopy," Laboratory Investigation, vol. 14/6, pp. 801–807, 1965.

Anisotropic Diffusion



Anisotropic Diffusion



Anisotropic diffusion

The original article (P. Perona, J Malik (July 1990). "Scale-space and edge detection using anisotropic diffusion". IEEE Tr. PAMI, 12 (7): 629–639.) can be found here: http://image.diku.dk/imagecanon/material/PeronaMalik1990.pdf

Create a function that:

- realizes anisotropic diffusion,
- parameters of this function should be as follows:
  - *input1:* image matrix,
  - *input2:* iteration number,
  - *input3:* $\boldsymbol{K}$,
  - *input4:* λ,
  - *output:* processed image,
- it should work with grayscale images (check if this holds),
- although formulas (7), (8), (10) and the two different versions of function $\boldsymbol{g}$ (at the bottom right of page 633) are enough to the implementation, I would like to recommend to **read the first 5 sections** of the article.

Anisotropic diffusion - continued

Create a script that:

- reads in this image: AndreKertesz_Paris_ManOnBicycle_part.jpg,
- converts it to grayscale,
- calls your anisotropic diffusion function on it (parameters eg. iternum: 200, K: 4, λ: 0.1),
- displays the input image and the diffused image.

Anisotropic diffusion - continued

My result with the exponential g-function:

Anisotropic diffusion - continued

My result with the "*1/(1+...)*" type *g*-function:

Anisotropic diffusion - continued

Observe how the different regions have changed:

- the originally homogeneous parts (within one region) diffused a lot, but
- the regions around edges (higher spatial frequency --- parts across regions) diffused less (what's more, they became sharper).