Basic Image Processing Algorithms

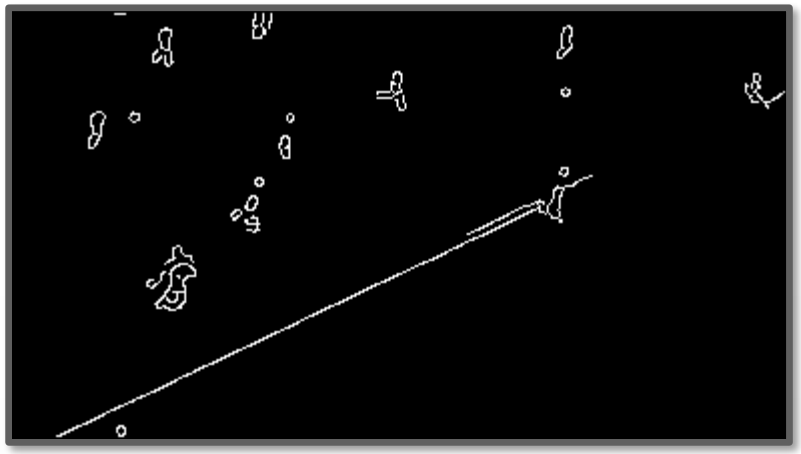
PPKE-ITK

Lecture 3.

Line detection with Hough Transfrom

Hough Transformation

- An example of Canny edge detector…



- …where straight lines are not detected perfectly.
- The objective of the Hough transformation is to find the lines on a binary image, from fragments/points of the line.

Finding lines in an image

- Option 1:
  - Search for the line at every possible position/orientation
  - What is the cost of this operation?
- Option 2:
  - Use a voting scheme: Hough transform

Finding lines in an image

- The basic idea:
  - A line can be written in the following form:

$$y = mx + b$$

where ***m*** is the slope of the line and ***b*** is the y-intercept.

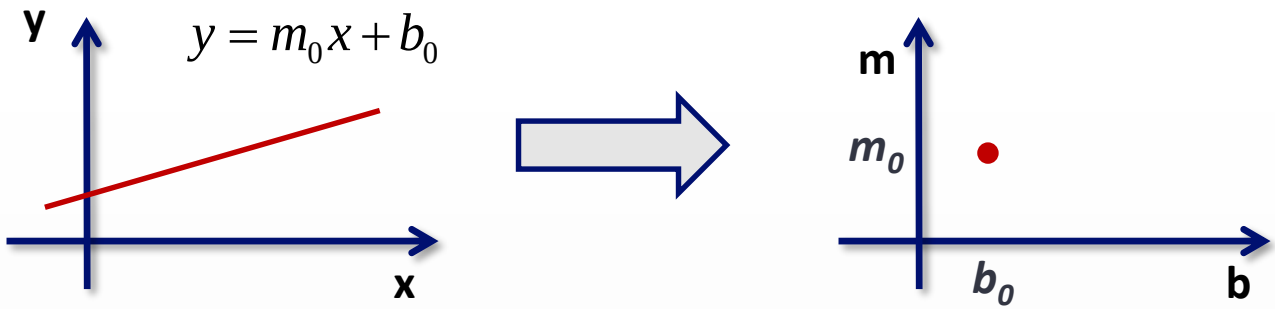
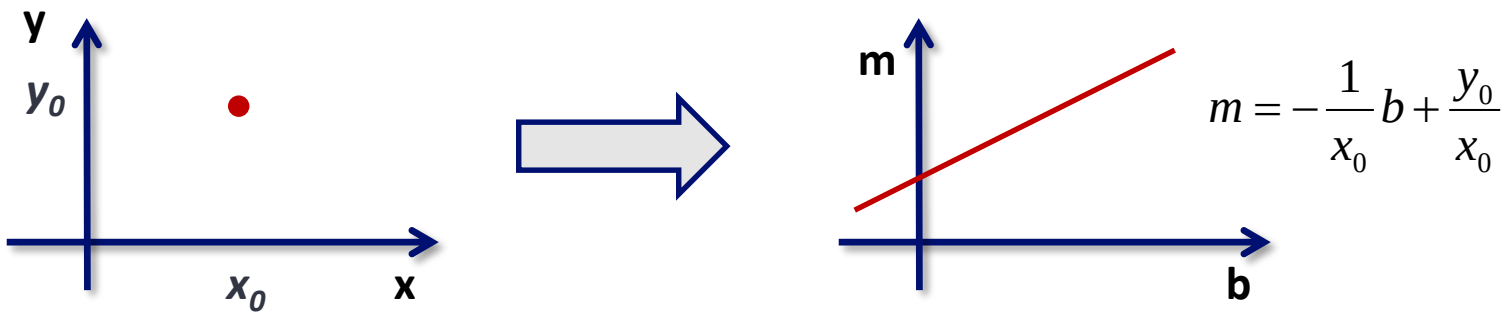


image space

m-b space of lines

- Connection between image ***(x,y)*** and the ***(m,b)*** spaces
  - A line in the image corresponds to a point in "m-b" space
  - To go from image space to (m-b) space:
    - given a set of points ***(x,y)***, find all ***(m,b)*** such that ***y = mx + b***

Finding lines in an image

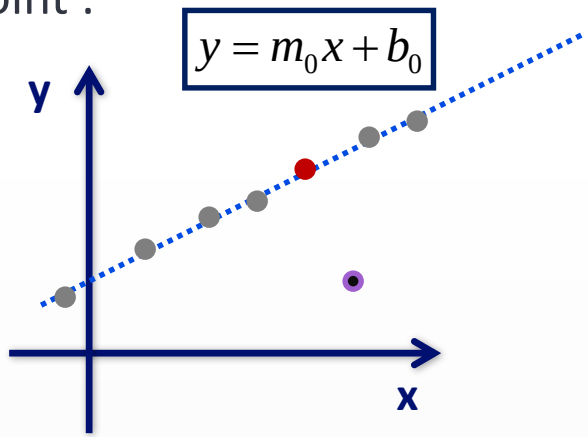


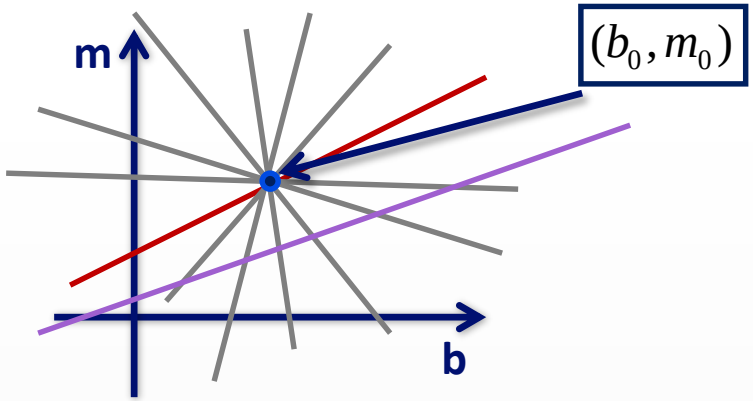
- Connection between image ***(x,y)*** and ***(m,b)*** spaces
  - What does a point ***(x_0, y_0)*** in the image space map to?
    - For a fixed ***$y = y_0$, $x = x_0$*** point in the image space, we get a line in the (m, b) space with a slope ***$-1/x_0$*** and an m-intercept: ***y_0/x_0*** :

$$m = -\frac{1}{x_0} b + \frac{y_0}{x_0}$$

Hough Transformation

- The basic idea:
  - For the points that lie on the same line in the Euclidian space, their corresponding line in the parameter space will cross each other in one point :

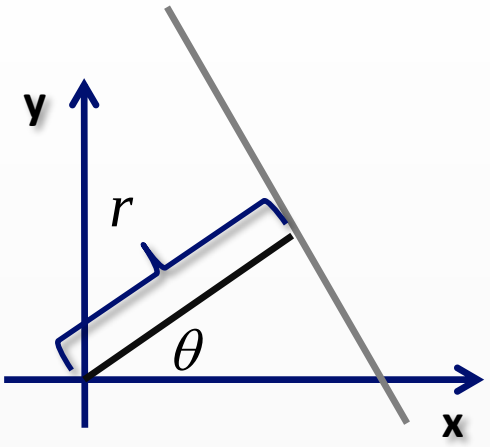




  - This point will be ***m=m$_0$*** and ***b=b$_0$***, the slope and intercept of the line in the image space. ⇨ We have the equation of the line!
- But, there is a problem with this equation of the line: vertical lines cannot be described (their slope would be infinite).

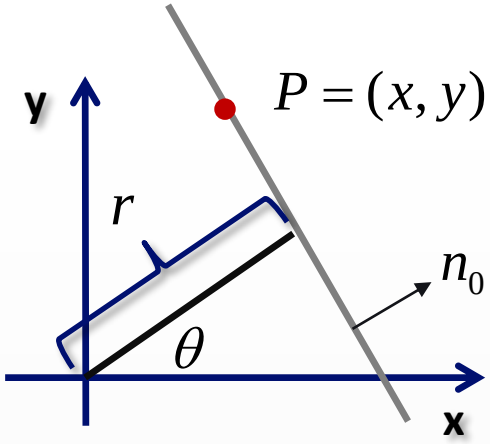
Hough Transformation

- To be able to describe all possible lines with two scalar parameters, we will use a **polar representation** of the line
- Each line is described by (r, θ) instead of (m, b), where
  - *r* is the perpendicular distance from the line to the origin
  - θ is the angle this perpendicular makes with the x axis



Hough Transformation

- Mathematical basis for using the **polar equation** of the line is the Hesse normal form*: $0 = P \cdot n_0 - r$



$P = (x, y)$ arbitrary point of the line

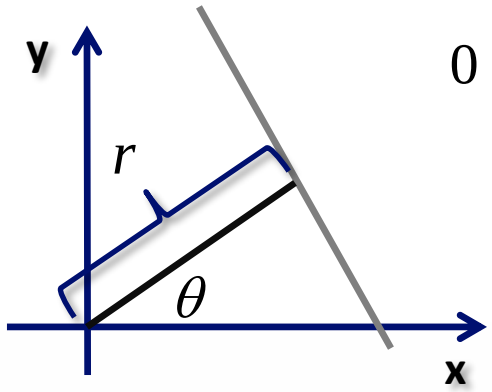
$n_0 = (\cos\theta, \sin\theta)$ normal vector of the line

- *r:* perpendicular distance from the line to the origin
- θ: the angle this perpendicular makes with the x axis

* https://en.wikipedia.org/wiki/Hesse_normal_form

Hough Transformation

- Hesse normal form based polar equation of the line:



$$0 = P \cdot n_0 - r = (x, y) \cdot (\cos\theta, \sin\theta) - r$$

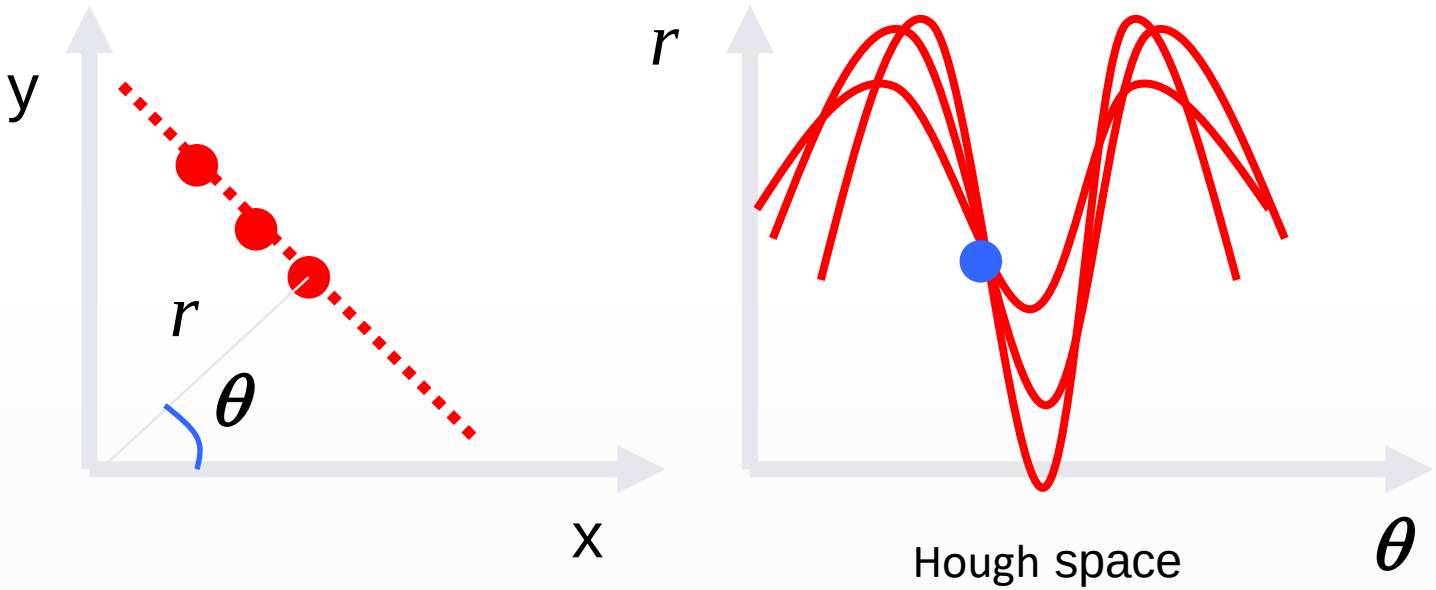
$$r = x \cdot \cos\theta + y \cdot \sin\theta$$

- The (r, θ) parameter space is called Hough space.
- A point in the Euclidian space is a sinusoid in the Hough space, described by the following equation:

$$r(\theta) = x \cdot \cos\theta + y \cdot \sin\theta$$

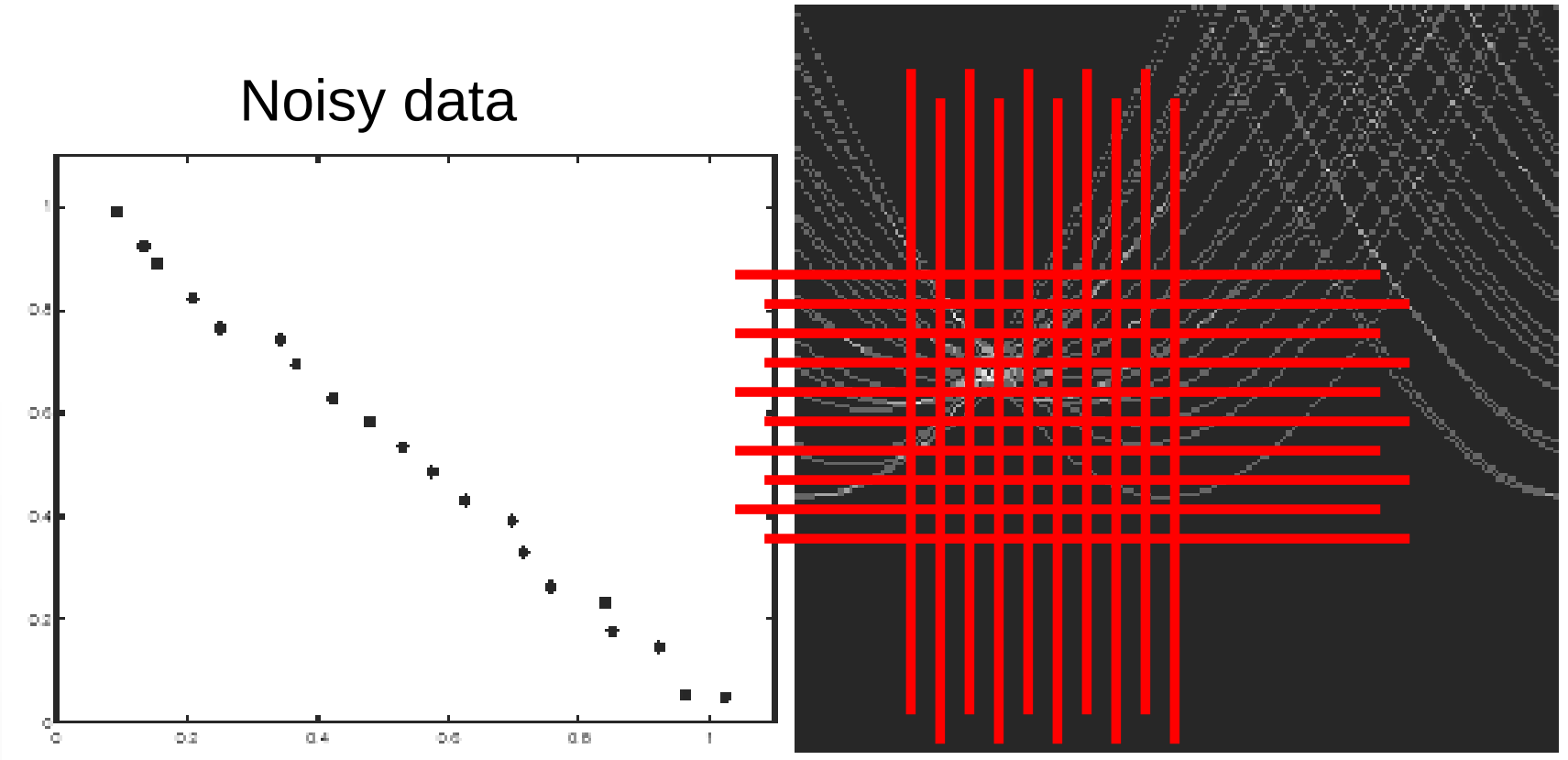
Hough Transformation

- All the sinusoid curves of the points in one line in the Euclidian space, cross each other in one point in the Hough space.



$$r(\theta) = x \cdot \cos\theta + y \cdot \sin\theta$$

Hough Transformation



Issue: Grid size needs to be adjusted…

Hough transform algorithm

- Basic Hough transform algorithm
  1. for all r, θ: initialize H[r, θ]=0
  2. for each edge point I[x,y] in the image

     for θ = 0 to 180

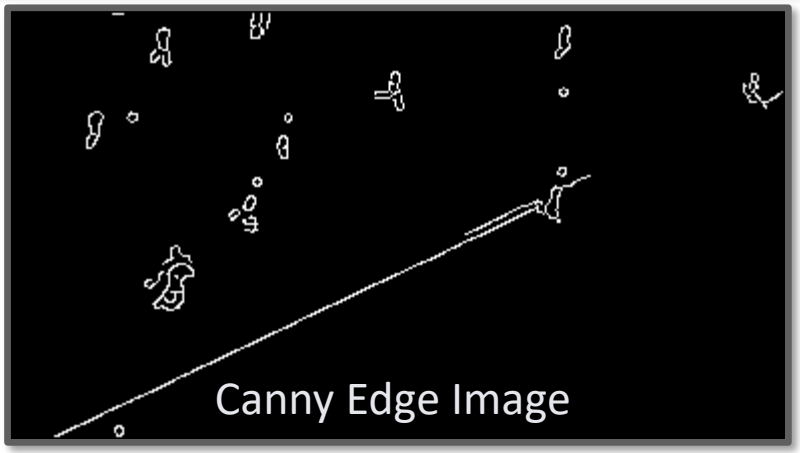
     $$r = x \cdot \cos\theta + y \cdot \sin\theta$$

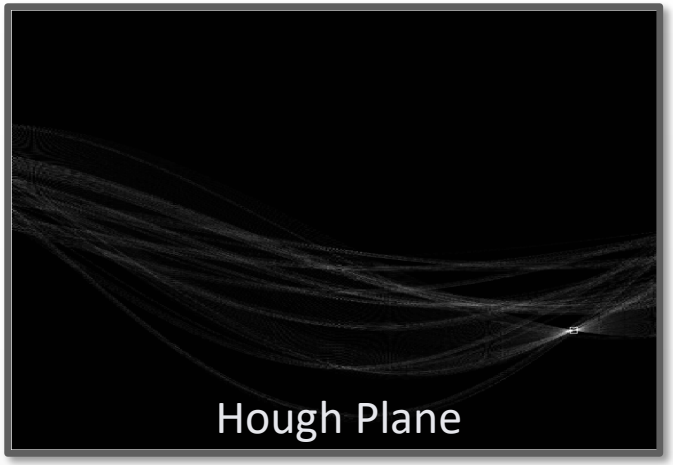
     H[r, θ] += 1
  3. Find the value(s) of (r, θ) where H[r, θ] is maximum
  4. The detected line in the image is given by

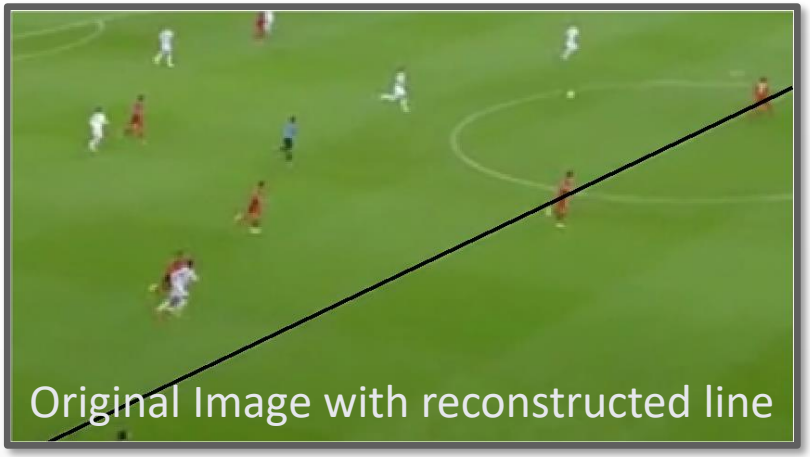
$$r = x \cdot \cos\theta + y \cdot \sin\theta$$

- What's the running time (measured in # votes)?

Hough Transformation







Extensions

- Extension 1: Use the image gradient
  1. same
  2. for each edge point I[x,y] in the image
     compute unique (r, θ) based on local image gradient at (x,y)
     H[r, θ] += 1
  3. same
  4. same

- Extension 2
  - give more votes for stronger edges

- Extension 3
  - change the sampling of (r, θ) to give more/less resolution

- Extension 4
  - The same procedure can be used with circles, squares, or any other shape

Hough demos

- Lines, circles and ellipses: http://dersmon.github.io/HoughTransformationDemo/
- Circle : http://www.markschulze.net/java/hough/

Image Enhancement

What is Image Enhancement?

- **Image enhancement** is the manipulation or transformation of the image to improve the visual appearance or to help further automatic processing steps.
- There is no general theory behind it, the result is highly application dependent and subjective.
  - e.g. in many cases the goal is to improve the quality for human viewing (Medical Imaging, Satellite Images)
- Enhancement is closely related to image recovery.
- Examples:
  - Contrast enhancement
  - Edge enhancement
  - Noise removal/smoothing

Types of Image Enhancement

- There are two main categories:
  - Spatial Domain Methods
  - Frequency Domain Methods
- In the Spatial Domain we are directly manipulating pixel values, through..
  - Point-wise Intensity Transformation
  - Histogram Transformations
  - Spatial Filtering
    - LSI (*Linear Shift-Invariant*)
    - Non-Linear
  - etc.

The Histogram of an Image

- **Histogram**:

  h(k) = the number of pixels on the image with value *k.*

Original Image*

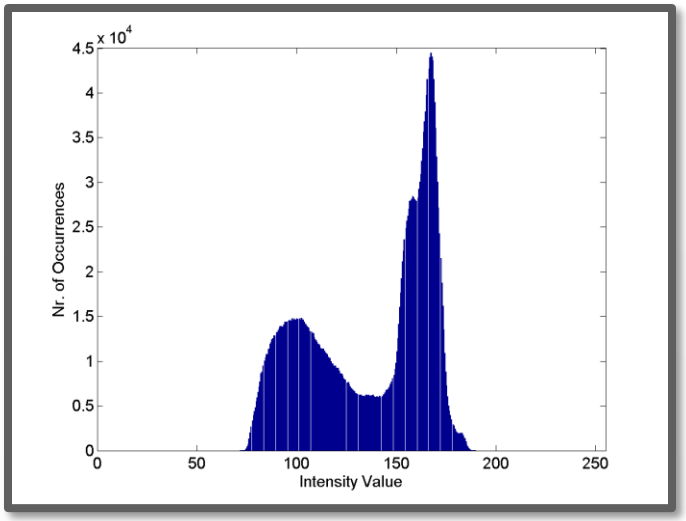


Image Histogram

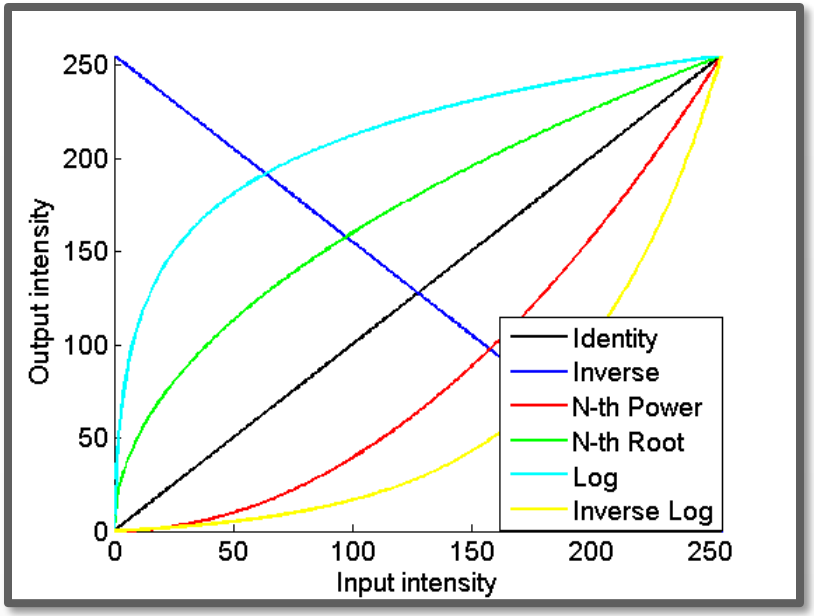
- The histogram normalized with the total number of pixels gives us the ***probability density function*** of the intensity values.

* Modified version of Riverscape with Ferry by Salomon van Ruysdael (1639)

Point-wise Intensity Transformation

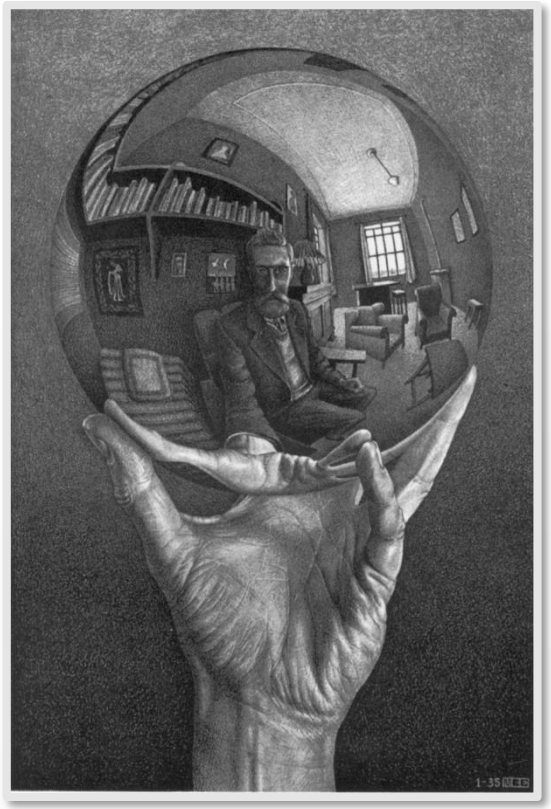
- Point wise transformations are operating directly on pixel values, independently of the values of its neighboring pixels.
- We can describe the transformation as follows:
  - Let *x* and *y* be two grayscale images, and let *T* be a point-wise image enhancement transformation that transforms *x* to *y*:

$$y(n_1, n_2) = T\left[x(n_1, n_2)\right]$$

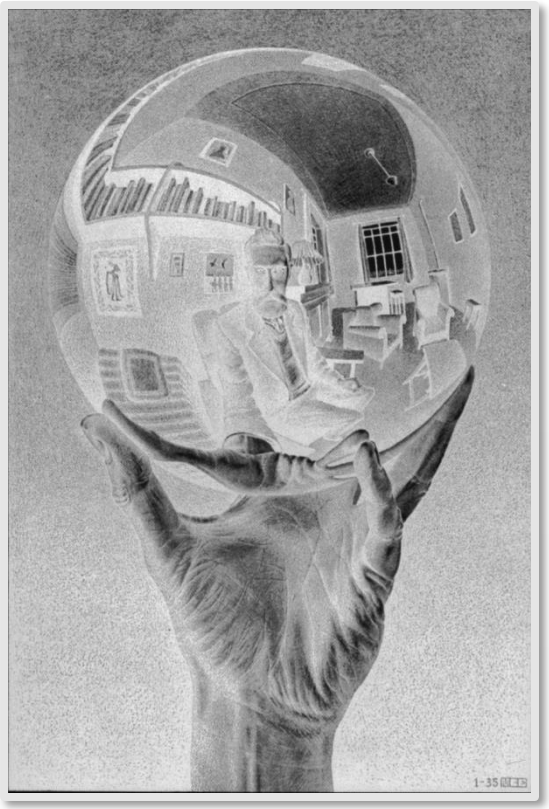


Point-wise Intensity Transformation

- **Inverse transformation**: $y(n_1, n_2) = 255 - x(n_1, n_2)$



Original Image*



Inverse Image

*Hand with Reflecting Sphere by M. S. Escher (1935)

Point-wise Intensity Transformation

- **Log transformation**: $y(n_1, n_2) = c \cdot \log\big(x(n_1, n_2) + 1\big)$
  - Expands low and compresses high pixel value range

Original Image*

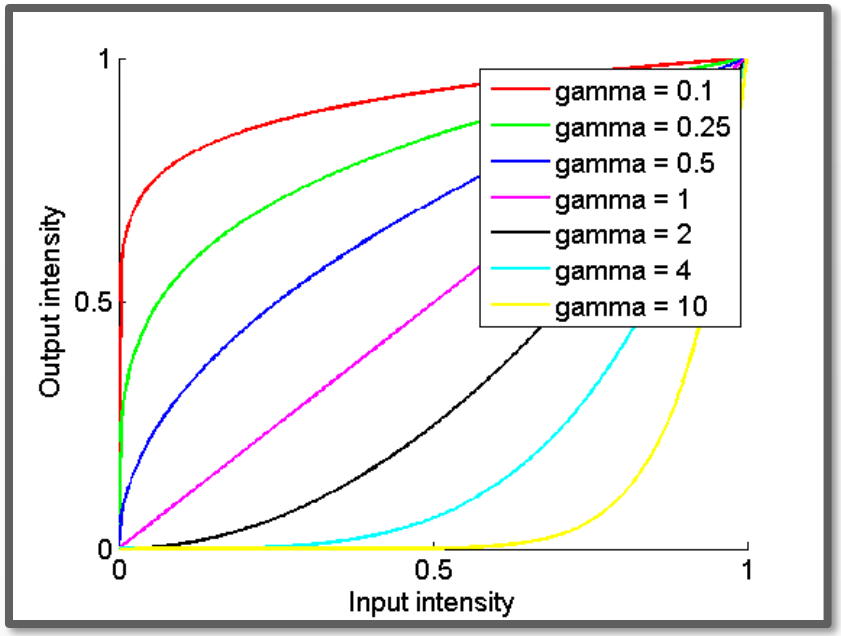
Log Image

Log Image
after histogram stretching

* Abbaye du Thoronet by Lucien Hervé (1951)

Point-wise Intensity Transformation

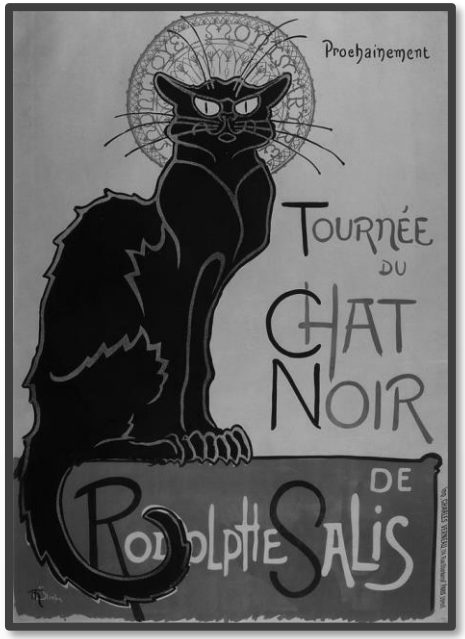
- **Power-law transformation**: $y(n_1, n_2) = c \cdot x(n_1, n_2)^{\gamma}$
  - Commonly referred to as ***gamma transformation***
  - Originally it was developed to compensate the input-output characteristics of CRT displays.
  - The expended/compressed region depends on γ:



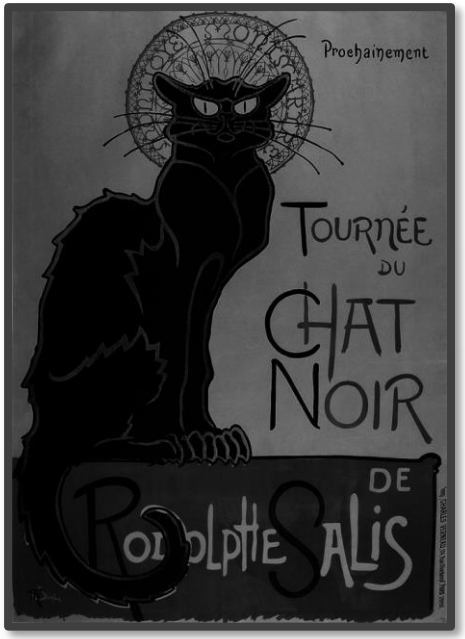
Here *c = 1*

Point-wise Intensity Transformation

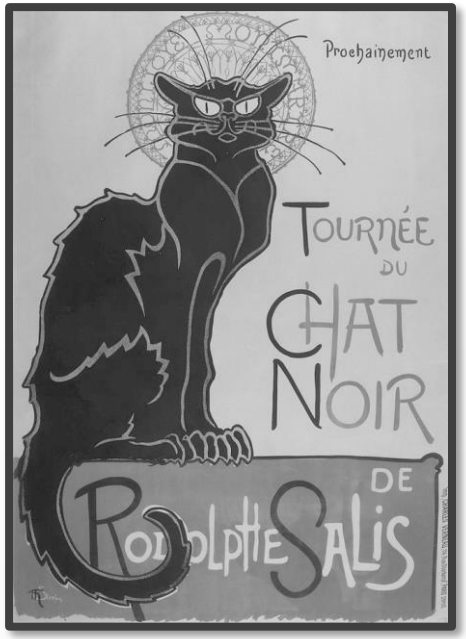
- **Power-law transformation**: $y(n_1, n_2) = c \cdot x(n_1, n_2)^{\gamma}$



Original Image*



γ = 2

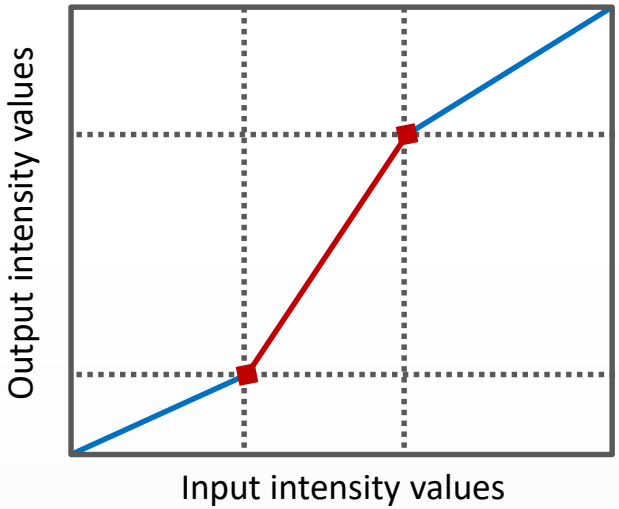


γ = 0.5

* Le chat Noir, Poster of Théophile Steinlen (1896)

Dynamic Range Expansion

- Piecewise linear expansion/compression of predefined intensity ranges:

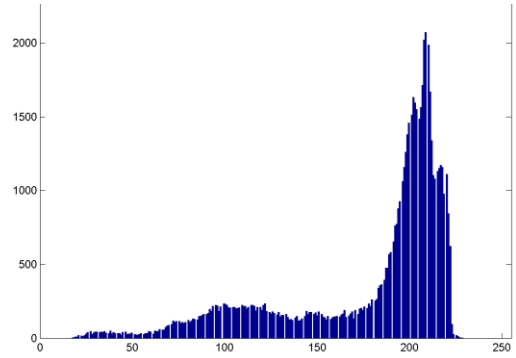


  - The red intensity range was expanded, while the blue ranges were compressed.

Dynamic Range Expansion

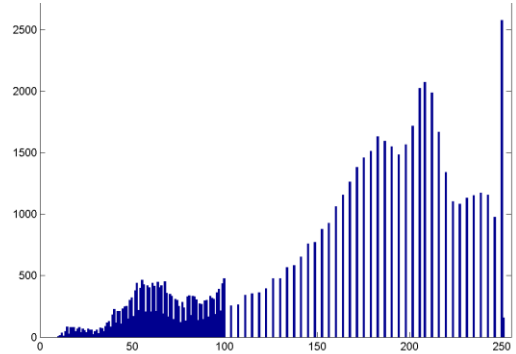
Original Image

Original Histogram



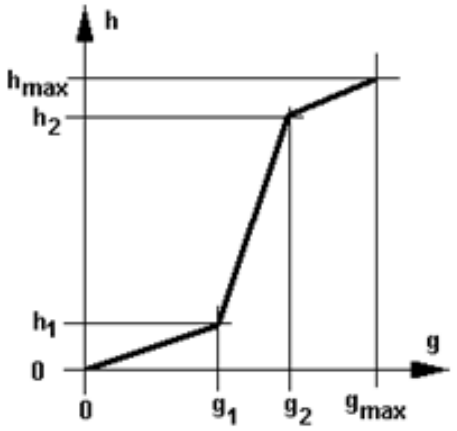
Modified Image

Histogram of the Modified Image



Dynamic Range Expansion

- Example: extracting the intensity values from the $[g_1, g_2]$ interval to a wider $[h_1, h_2]$ domain
  - Enhanced contrast in the selected region, details are better observable and distinguishable.
  - In the remaining image regions the contrast decreases



Histogram Transformations

- **Histogram Stretching**:
  - Based on the histogram we can see that the image does not use the whole range of possible intensities:
    - Minimum intensity level: 72
    - Maximum intensity level: 190
  - With the following transformation we can stretch the intensity values so they use the whole available range:

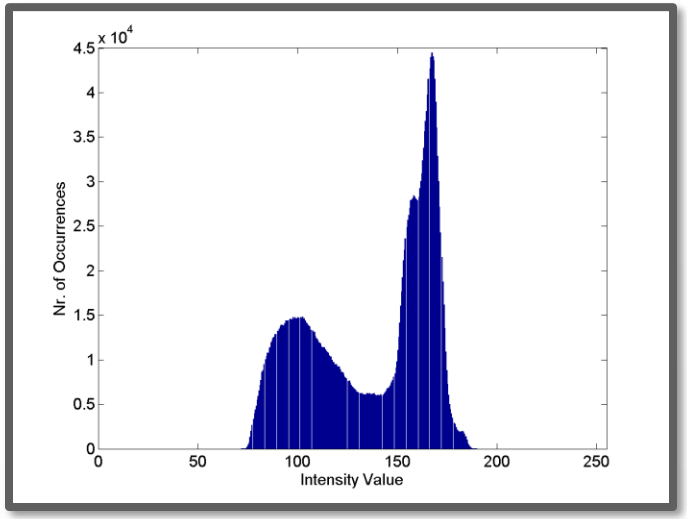


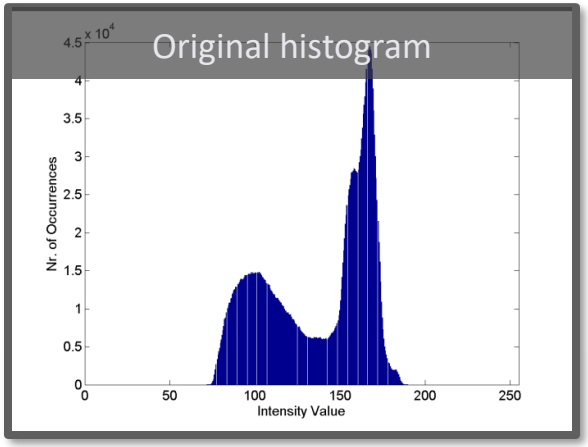
Image Histogram

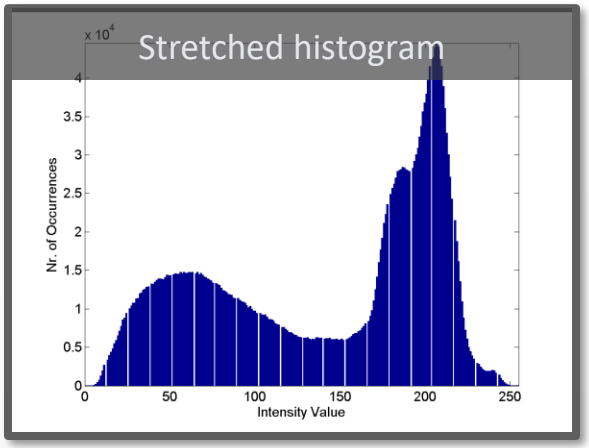
$$y(n_1, n_2) = \frac{255}{x_{\max} - x_{\min}} \cdot (x(n_1, n_2) - x_{\min})$$

$$x_{\max} = \max_{n_1, n_2}(x(n_1, n_2)) \qquad x_{\min} = \min_{n_1, n_2}(x(n_1, n_2))$$

Histogram Transformations

- **Histogram Stretching**:





Histogram Transformations

- Histogram stretching with various transfer functions:
  - Linear:

$$y(n_1,n_2)=\frac{255}{x_{\max}-x_{\min}}\cdot(x(n_1,n_2)-x_{\min})=255\cdot\frac{x(n_1,n_2)-x_{\min}}{x_{\max}-x_{\min}}$$

  - Quadratic:

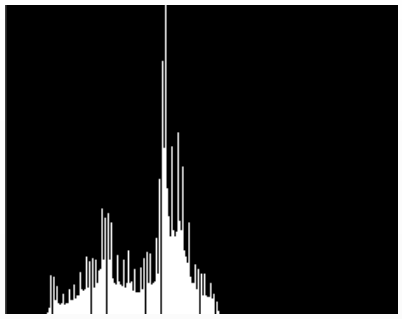
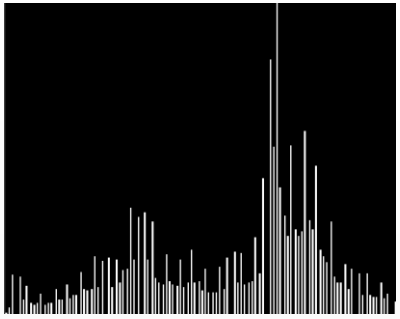
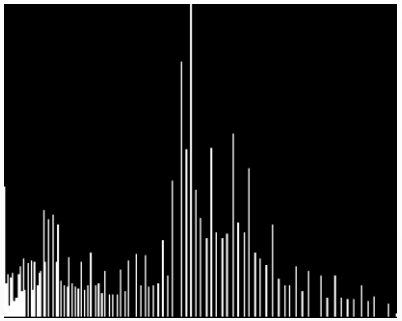
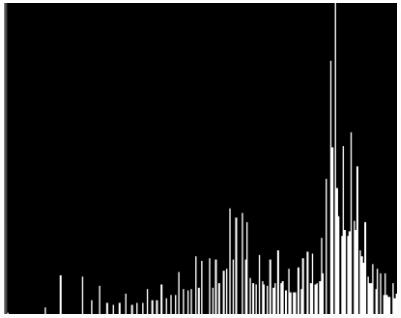
$$y(n_1,n_2)=255\cdot\left(\frac{x(n_1,n_2)-x_{\min}}{x_{\max}-x_{\min}}\right)^2$$

  - Square root

$$y(n_1,n_2)=255\cdot\sqrt{\frac{x(n_1,n_2)-x_{\min}}{x_{\max}-x_{\min}}}$$

Histogram stretching - results



original

linear f()

quadratic

square root

Histogram stretching - results

original

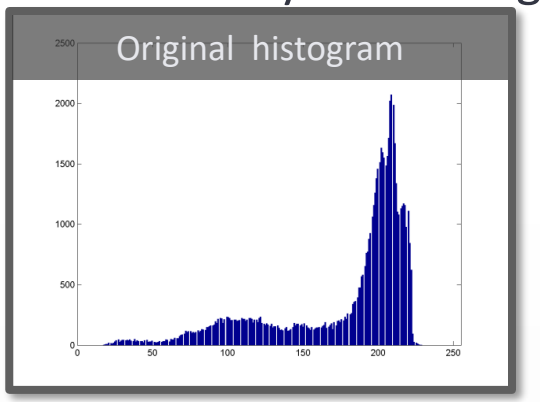
linear f()

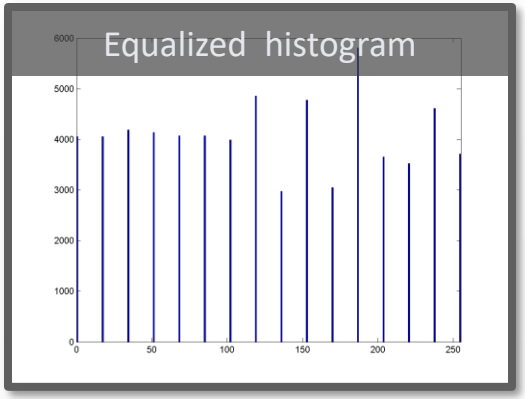
quadratic

square root

Histogram Transformations

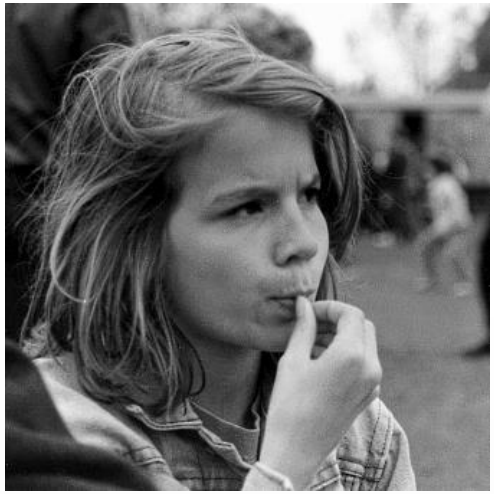
- **Histogram Equalization**:
  - The goal is to increase the contrast, by distributing the occurrences of the intensity values evenly through the entire dynamic range.

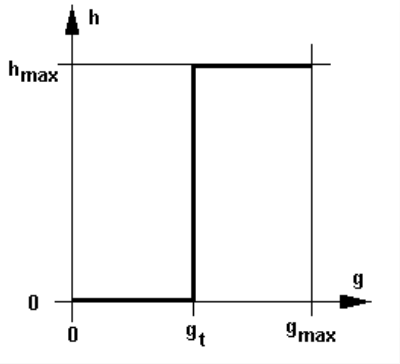




Histogram equalization background

- Simple thresholding





- For different g_t values

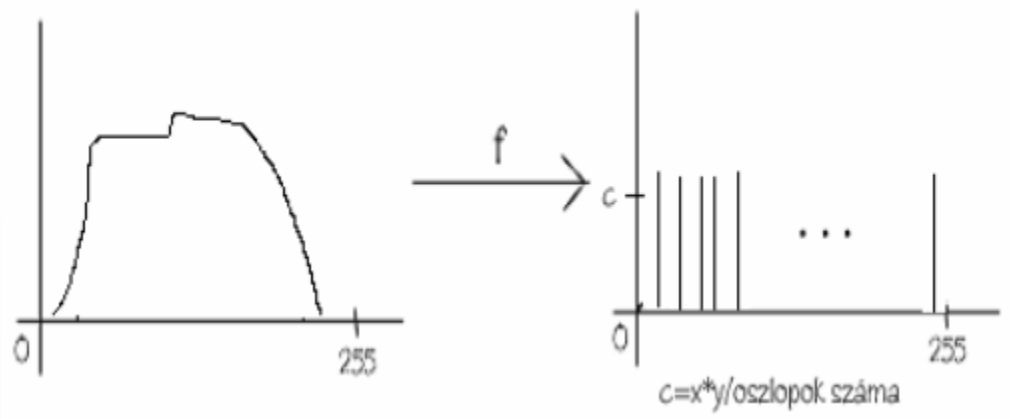
Optimal threshold value

- Task: converting a grayscale image to binary (black&white). What is the optimal threshold value?
  - A possible good solution is to prescribe that the number of black and white pixels should be approximately the same in the output image.
    - The g_t threshold value can be calculated from the histogram, (P is the total number of pixels):

$$\sum_{i=0}^{g_t} h[i] \approx \sum_{i=g_t+1}^{255} h[i] \approx \frac{P}{2}$$

Generalization: histogram equalization

- **Goal**: contrast enhancement
- Transform: step (staircase) function. The number of columns determines number of color (intensity) values appearing in the output, (e.g. number of columns=16, 32, 64, etc.).



Histogram equalization – c output values

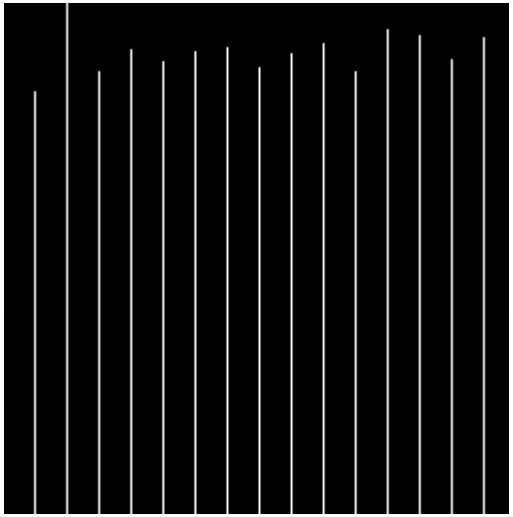
- Goal: determining the $t_0=0$, t_1, … t_{c-1},$t_c=255$ dividing points, where:

$$\sum_{i=0}^{t_j} h[i] \approx P \cdot \frac{j}{c} \qquad j \in \{1 \ldots c\}$$

  - c is the number of different gray levels in the output image (c=2 for thresholding, but it can also be 16, 32, … 256 as well)
  - P is the total number of pixels again.

Histogram equalization - result

16 level ouptut



Histogram of the output image

Histogram Transformations

- Adaptive Histogram Equalization:
  - applies histogram equalization on parts of the image (called tiles) independently
  - Use post processing to reduce artifacts at the borders of the tiles.

Original image

Image after CLAHE

[1] Zuiderveld, Karel. "Contrast Limited Adaptive Histograph Equalization." *Graphic Gems IV*. San Diego: Academic Press Professional, 1994. 474–485.

Spatial Filtering

- **Smoothing:**
  - Reduce the noise that may corrupt the image.
- A few noise types we will work with:
  - Impulse noise, (aka salt and pepper noise)
  - Additive Gaussian Noise

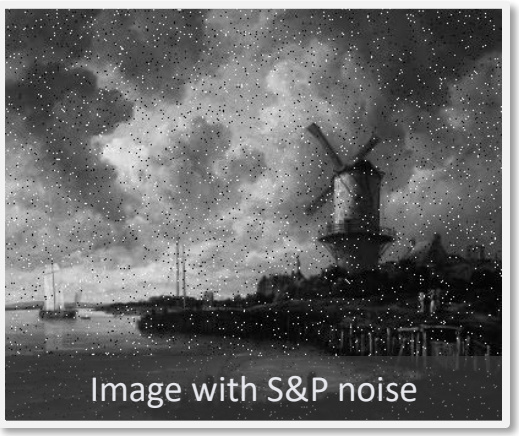
Additive Gaussian Noise

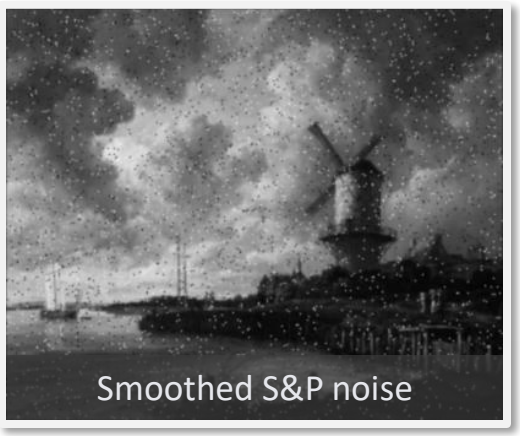
Impulse Noise

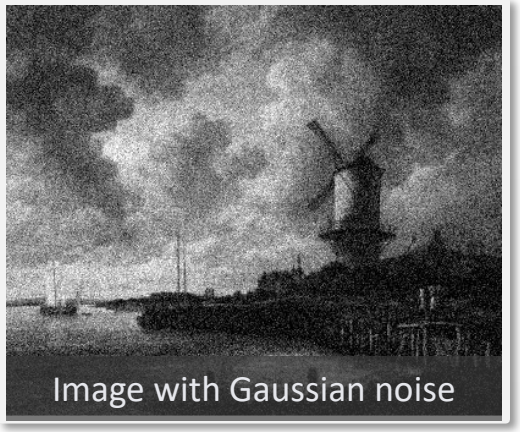
The windmill at Wijk bij Duurstede by Jacob van Ruisdael (1670)

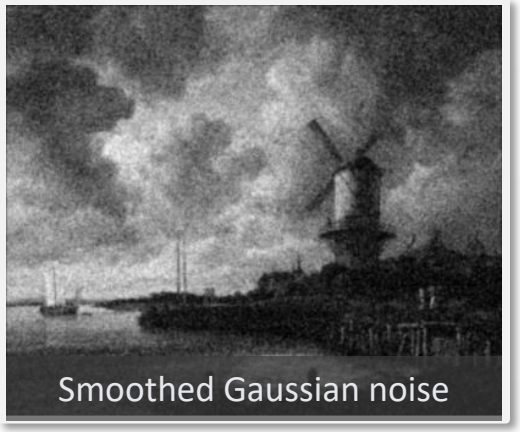
Spatial Filtering

- **Gaussian Smoothing:**
  - With $\sigma=0.75$



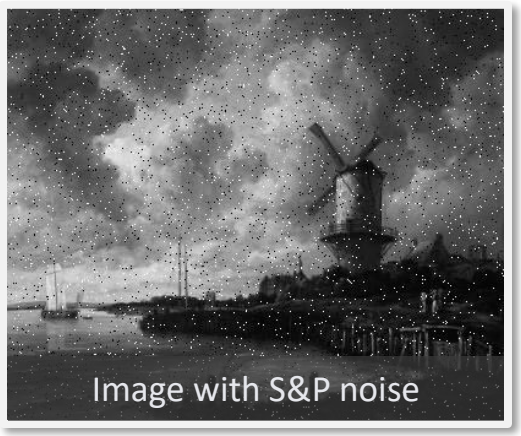


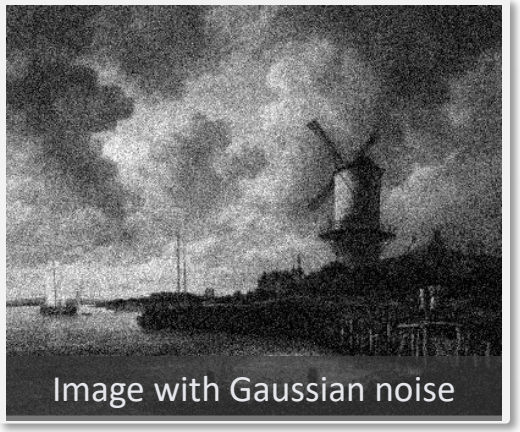




Spatial Filtering

- **Gaussian Smoothing:**
  - With $\sigma=1.5$





Spatial Filtering

- **Spatially Adaptive Noise Smoothing:**
  - The smoothing takes into account the local characteristics of the image:

$$y(n_1,n_2)=\left(1-\frac{\sigma_n^2}{\sigma_l^2}\right)\cdot x(n_1,n_2)+\frac{\sigma_n^2}{\sigma_l^2}\,\bar{x}(n_1,n_2)$$

$$\sigma_l^2(n_1,n_2)=\sum_{(n_1,n_2)\in N}\sum\left(x(n_1,n_2)-\bar{x}(n_1,n_2)\right)^2$$

Local variance of the image

$$\bar{x}(n_1,n_2)=\frac{1}{|N|}\sum_{(n_1,n_2)\in N}\sum x(n_1,n_2)$$

Local average of the image

Variance of the noise: either known a priori, or has to be measured

Spatial Filtering

- **Spatially Adaptive Noise Smoothing:**


Original image*

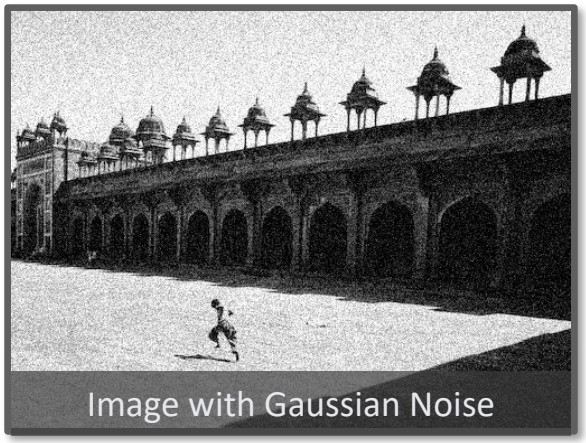
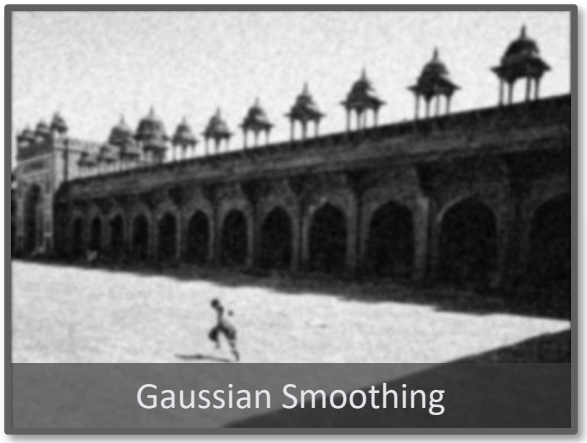
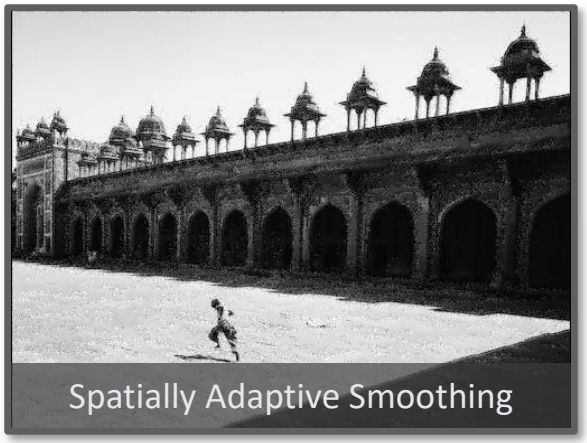


Image with Gaussian Noise


Gaussian Smoothing


Spatially Adaptive Smoothing

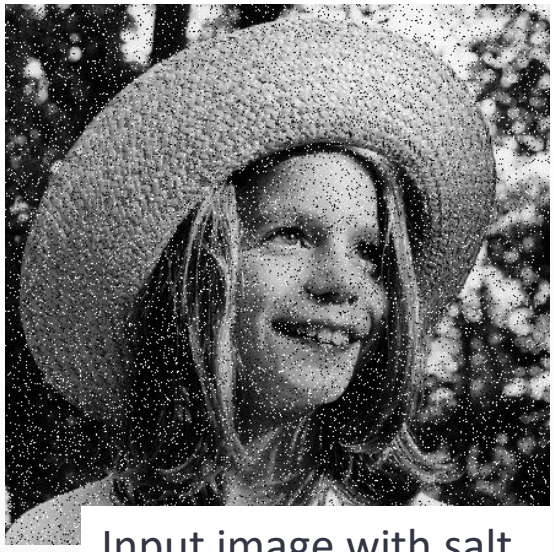
* Fatepuhr Sikri, Inde by Lucien Hervé (1955)

Spatial Filtering

- **Rank filter:**
  1. Consider the actual pixel and its neighborhood (e.g. 3×3=9 pixel sized window),
  2. Sort the observed pixel values according to gray level,
  3. Take the *k-th* value from this row as the new pixel value
- **Median filter**: k is the middle pixel value in the row:
  - $k=[(2W+1)^2-1]/2$, if W is the half side size of the neighborhood
- Non-Linear filter

Spatial Filtering

- **Median filter:** replaces each pixel with the *median value* of its analyzed neighborhood. (Median value: the center element of sorted values)
  - Very effective against impulse („salt and pepper”) noise:



Input image with salt and pepper noise

Blur with convolution

Median filter

Spatial Filtering

- **Median filter:** replaces each pixel with the *median value* of its analyzed neighborhood. (Median value: the center element of sorted values)
  - Very effective against impulse („salt and pepper”) noise:


Original image

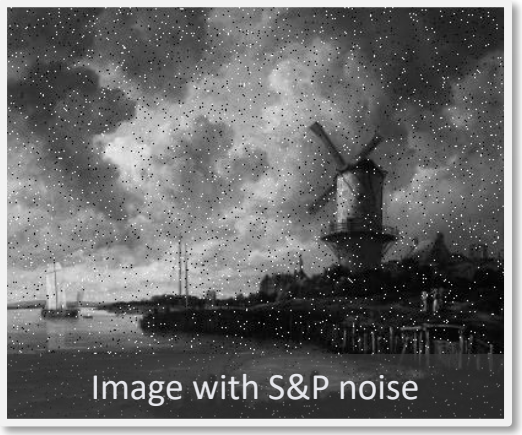


Image with S&P noise


Median filtered S&P noise

  - Not so effective against Gaussian noise.

Spatial Filtering

- **Order statistic filtering:**
  - Based on the sorted pixel intensity levels in the analyzed neighborhood.
  - If after sorting…
    - we take the middle element, we get back the median filter.
    - We take the maximum element to filter „pepper” and min to filter „salt” noise.

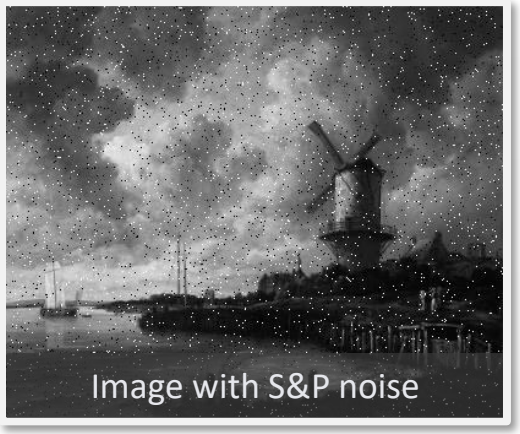
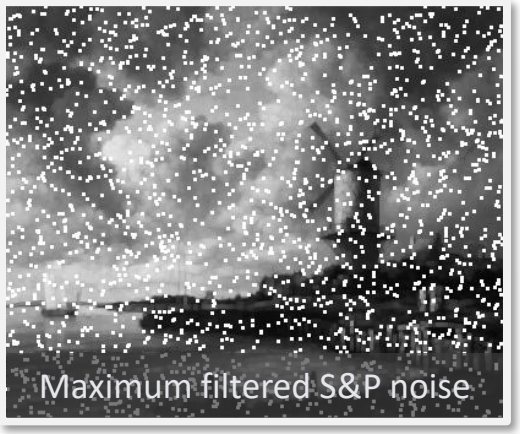


Image with S&P noise


Maximum filtered S&P noise


Minimum filtered S&P noise

- But max filter will highlight „salt”, while min filter will highlight „pepper”.

Spatial Filtering

- **Order statistic filtering:**
  - Mid-point filtering:
    - works well on Gaussian or uniform noise

$$y(n_1, n_2) = \frac{1}{2}\left(\max_{(m_1,m_2)\in N} \{x(m_1, m_2)\} + \min_{(m_1,m_2)\in N} \{x(m_1, m_2)\} \right)$$

  - Alpha-trimmed mean filter:

$$y(n_1, n_2) = \frac{1}{|N| - \alpha} \sum_{(m_1,m_2)\in N_r} x(m_1, m_2)$$

    - Where N_r is a reduced neighborhood, not containing the lowest and highest α element of N.
    - If α = 0, we get back the arithmetic mean.
    - If α = |N|-1, we get back the median filter.

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: $\sigma_l(n_1, n_2) = \frac{1}{|N|}\sqrt{\sum\sum_{(n_1,n_2)\in N}(x(n_1, n_2) - \bar{x}(n_1, n_2))^2}$
- $\bar{x}$ is the local average: $\bar{x}(n_1, n_2) = \frac{1}{|N|}\sum\sum_{(n_1,n_2)\in N} x(n_1, n_2)$
- 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

Original Image

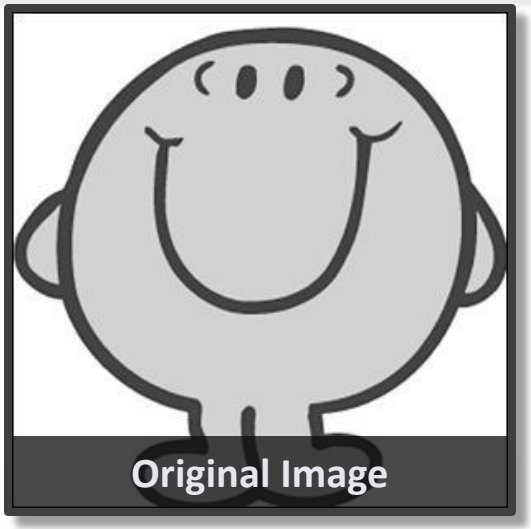
Image after applying Wallis operator

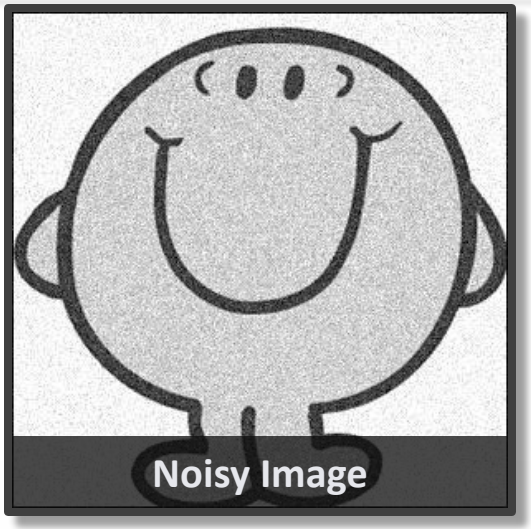
Anisotropic Diffusion

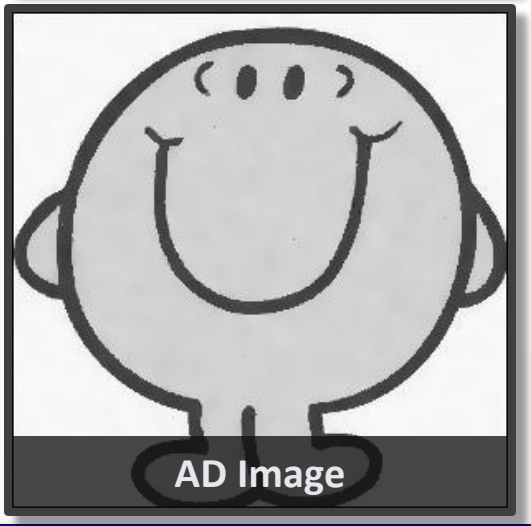
- The anisotropic diffusion is a technique aiming at reducing image noise without blurring significant parts of the image content.
- It was first proposed by D. Gábor in 1965 and later by Perona and Malik around 1990.
- ***Non-linear*** and ***space-variant*** transformation.
- The main idea is that 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.

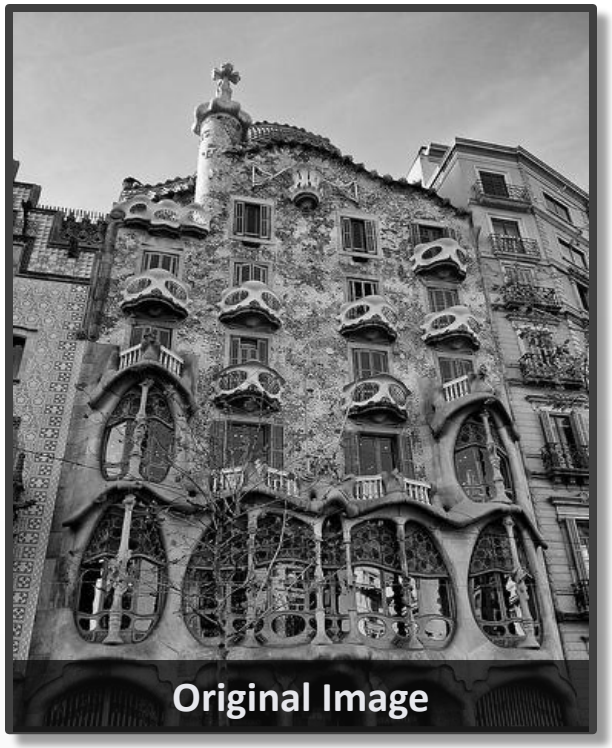
Anisotropic Diffusion

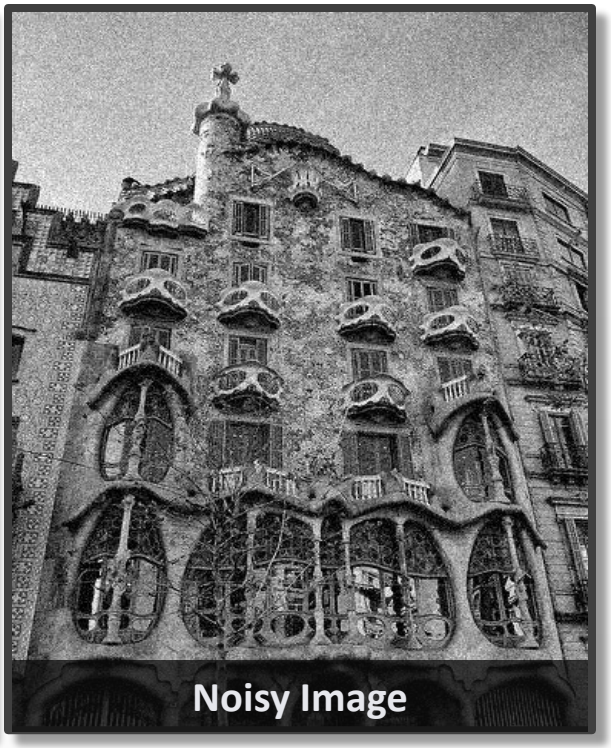






Anisotropic Diffusion





Total Variation Regularization

- Assumption:
  - The image is smooth inside the objects, with jumps across the boundaries.
  - The noise component has high variation.
- The goal of ***Total Variation based noise removal*** is to minimize the total variation of the image while keep the result as close to the original input image as possible.
- It was introduced by Rudin, Osher and Fatemi in 1992.

Rudin, L. I.; Osher, S.; Fatemi, E. (1992). "Nonlinear total variation based noise removal algorithms". *Physica D* **60**: 259–268

Total Variation Regularization

- TV of the output image y is defined as the integral of the absolute gradient of the signal:

$$\mathrm{V}(y)=\sum_{n_1}\sum_{n_2}\sqrt{\left|y(n_1+1,n_2)-y(n_1,n_2)\right|^2+\left|y(n_1,n_2+1)-y(n_1,n_2)\right|^2}$$

- On the other hand, we also measure the difference between the original image x and the output image y by L_2 norm E:

$$\mathrm{E}(\mathrm{x},\mathrm{y})=\sum_{n_1,n_2}\left(x(n_1,n_2)-y(n_1,n_2)\right)^2$$

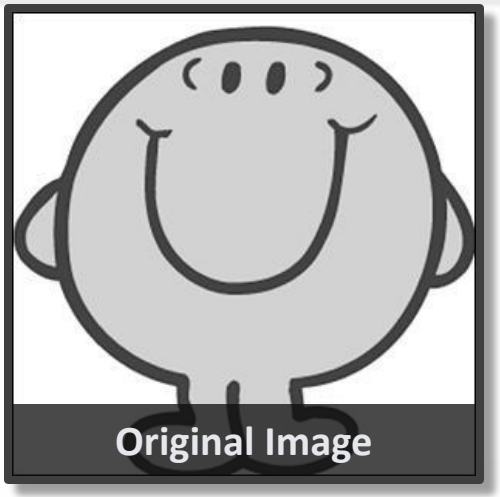
- The goal function for Total Variation based regularization:

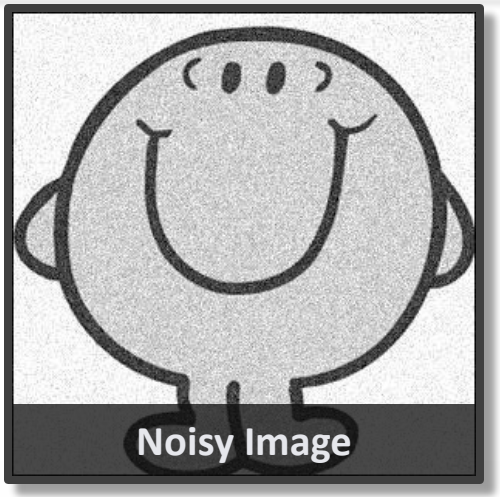
$$\hat{\mathrm{y}}=\arg\min_{y}\left[\mathrm{E}(\mathrm{x},\mathrm{y})+\lambda\mathrm{V}(y)\right]$$

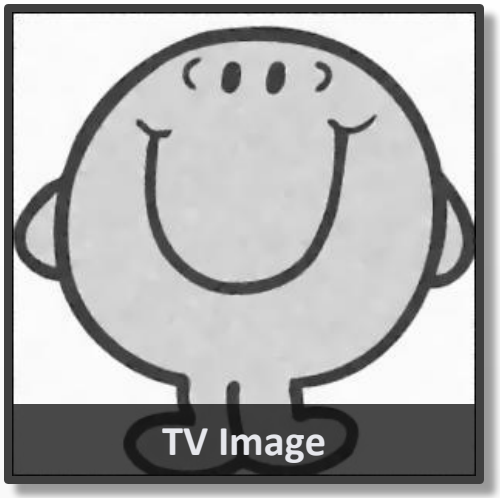
where λ is the regularization parameter.

Rudin, L. I.; Osher, S.; Fatemi, E. (1992). "Nonlinear total variation based noise removal algorithms". *Physica D* **60**: 259–268

Total Variation Regularization







Matlab Code: http://visl.technion.ac.il/~gilboa/PDE-filt/tv_denoising.html

Total Variation Regularization

Matlab Code: http://visl.technion.ac.il/~gilboa/PDE-filt/tv_denoising.html

Total Variation Regularization

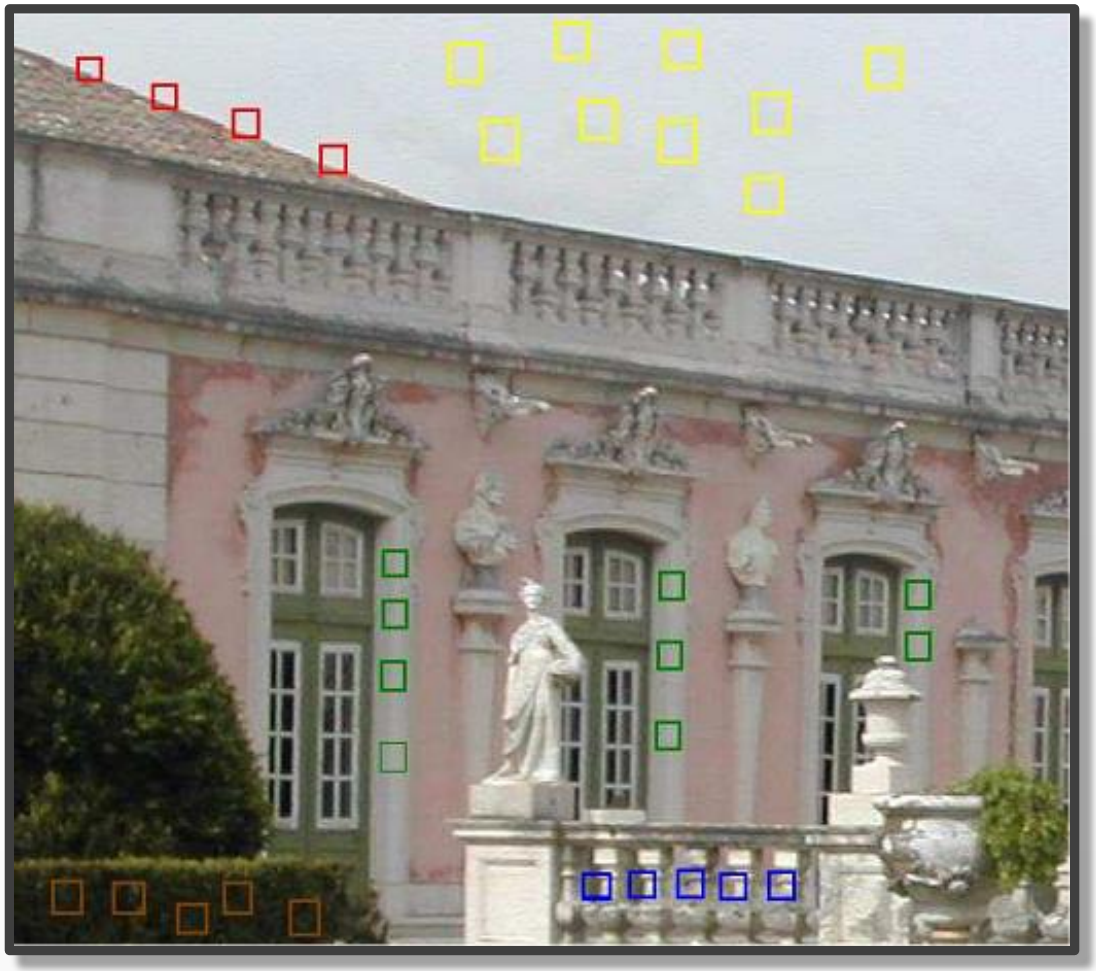
Matlab Code: http://visl.technion.ac.il/~gilboa/PDE-filt/tv_denoising.html

Non-Local Means Denoising

- The local smoothing methods aim at a noise reduction and at a reconstruction of the main geometrical configurations but not at the preservation of the fine structure, details and texture.
- ***High frequency image components are removed along with the noise, because they behave in all functional aspects as noise.***
- The NL-means algorithm tries to take advantage of the ***high degree of redundancy of any natural image***:
  - Every small window in a natural image has many similar windows in the same image.
  - The NL-means algorithm estimates the value of a pixel ***x*** as an average of the values of all the pixels whose Gaussian neighborhood looks like the neighborhood of ***x***.

More details and a demo is available here: http://www.ipol.im/pub/art/2011/bcm_nlm/

Non-Local Means Denoising



A. Efros and Th. K. Leung: Texture Synthesis by Non-parametric Sampling, IEEE ICCV, Corfu, Greece, September 1999

Non-Local Means Denoising

More details and a demo is available here: http://www.ipol.im/pub/art/2011/bcm_nlm/

Non-Local Means Denoising

More details and a demo is available here: http://www.ipol.im/pub/art/2011/bcm_nlm/