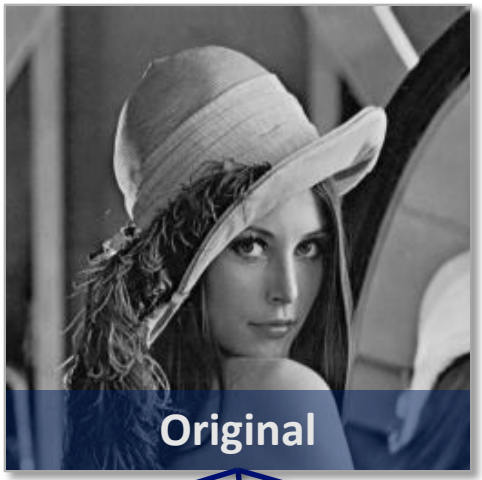
Basic Image Processing Algorithms

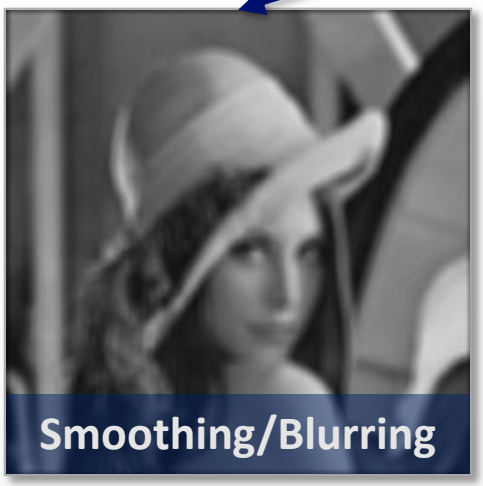
PPKE-ITK, 2015

Lecture 2.

2D Convolution

Examples


Original


Smoothing/Blurring


Edge Enhancement


Edge Detection

Mathematical background

- We look at the image as a 2D function:
- We can define different transformations:
  - Intensity value inversion:

$$y(n_1, n_2) = 255 - x(n_1, n_2)$$

  - Intensity shift with constant:

$$y(n_1, n_2) = x(n_1, n_2) + 100$$

  - Weighting:

$$y(n_1, n_2) = x(n_1, n_2)w(n_1, n_2)$$

  - Average on an *N* neighborhood:

$$y(n_1, n_2) = average\ \left(N\left(x(n_1, n_2)\right)\right)$$

- 2 important properties of the transformations we want to use on images: ***linearity*** and ***shift invariance***

Mathematical background

- Linearity:

$$T[x_1(n_1,n_2)+x_2(n_1,n_2)]=T[x_1(n_1,n_2)]+T[x_2(n_1,n_2)]$$

$$T[\alpha x(n_1,n_2)]=\alpha T[x(n_1,n_2)]$$

  - e.g.: weighting is linear, intensity inversion is non-linear

- Spatial Invariance:

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

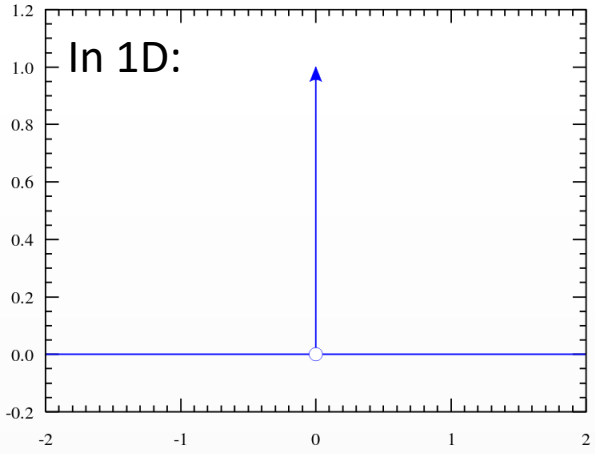
$$T[x(n_1-k_1,n_2-k_2)]=y(n_1-k_1,n_2-k_2)$$

  - e.g.: weighting is not SI, intensity inversion is SI
  - e.g.: averaging on neighborhood is LSI

Unit Impulse Function

- 2D Unit Impulse function (Delta function) on $\mathbb{Z}$ as follows:

$$\delta(n_1, n_2) = \begin{cases} 1 & \text{when } n_1 = 0 \text{ and } n_2 = 0 \\ 0 & \text{otherwise} \end{cases}$$



- For any 2D function $x(n_1,n_2)$:

$$x(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \delta(n_1 - k_1, n_2 - k_2) x(k_1, k_2)$$

Convolution

- ***Impulse response*** is the output of an LSI transformation if the input was the Delta function: $\delta(n_1, n_2) \rightarrow T \rightarrow h(n_1, n_2)$

  If *T* is an LSI system:

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

  Then we can define convolution as follows:

$$y(n_1, n_2) = x(n_1, n_2) * h(n_1, n_2) =$$

$$= h(n_1, n_2) * x(n_1, n_2) =$$

$$= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

Derivation of Convolution

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

$$= T\left[\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} x(k_1, k_2)\delta(n_1 - k_1, n_2 - k_2)\right] =$$

Linearity

$$= \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} x(k_1, k_2)T\left[\delta(n_1 - k_1, n_2 - k_2)\right] =$$

$$= \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} x(k_1, k_2)h(n_1 - k_1, n_2 - k_2)$$

Spatial Invariance

The Properies of Convolution

- Commutative:

$$f * g = g * f$$

- Associative:

$$f * (g * h) = (f * g) * h$$

- Distributive:

$$f * (g + h) = f * g + f * h$$

- Associative with scalar multiplication:

$$\alpha(f * g) = (\alpha f) * g$$

2D convolution in Practice

- In practice both the kernel and the image have finite size.

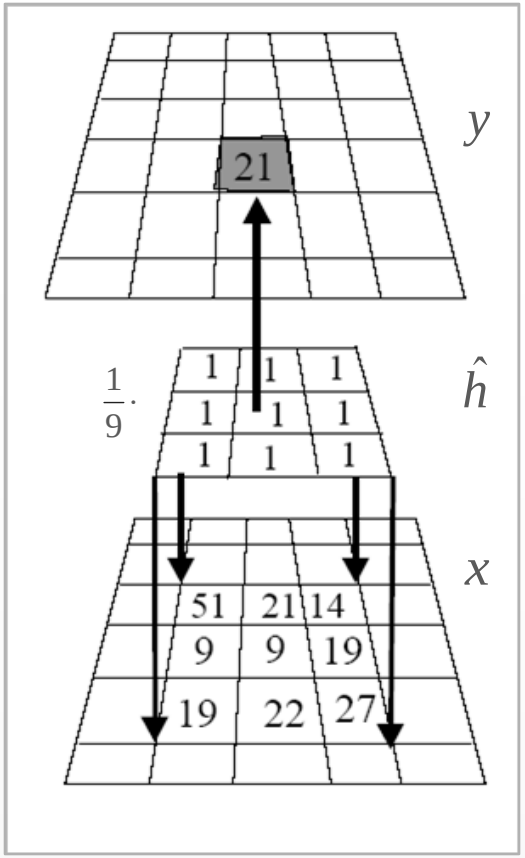
Let h and $\hat{h}$ be $(2r_1+1)\times(2r_2+1)$ sized kernels, where $\hat{h}$ is the $180°$ rotated version of h:

$$h=\begin{bmatrix} a_{-r_1,-r_2} & \cdots & a_{-r_1,r_2} \\ \vdots & \ddots & \vdots \\ a_{r_1,-r_2} & \cdots & a_{r_1,r_2} \end{bmatrix} \text{ and } \hat{h}=\begin{bmatrix} a_{r_1,r_2} & \cdots & a_{r_1,-r_2} \\ \vdots & \ddots & \vdots \\ a_{-r_1,r_2} & \cdots & a_{-r_1,-r_2} \end{bmatrix}$$

$$y(n_1,n_2)=\sum_{k_1=-r_1}^{r_1}\sum_{k_2=-r_2}^{r_2}x(k_1,k_2)h(n_1-k_1,n_2-k_2)=$$

$$=\sum_{k_1=-r_1}^{r_1}\sum_{k_2=-r_2}^{r_2}h(k_1,k_2)x(n_1-k_1,n_2-k_2)=$$

$$=\sum_{k_1=-r_1}^{r_1}\sum_{k_2=-r_2}^{r_2}\hat{h}(k_1,k_2)x(n_1+k_1,n_2+k_2)$$



Size of the Convolved Image

Convolutional Kernel

$$* \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

5x5

In general:

Size of the input image: ***(A x B)***

Size of the kernel: ***(C x D)***

Size of the output image: ***(A+C-1) x (B+D-1)***

Boundary Effects

- What happens at the border of the image?

Original image with the problematic area

Zero padding

Mirroring

Circular padding

Repeating border

Applications

- Possible application of convolution:
  - Smoothing/Noise reduction
  - Edge detection
  - Edge enhancement
- Depending on the task the *sum of the elements of the kernel matrix* can be different:
  - 1: smoothing, edge enhancement

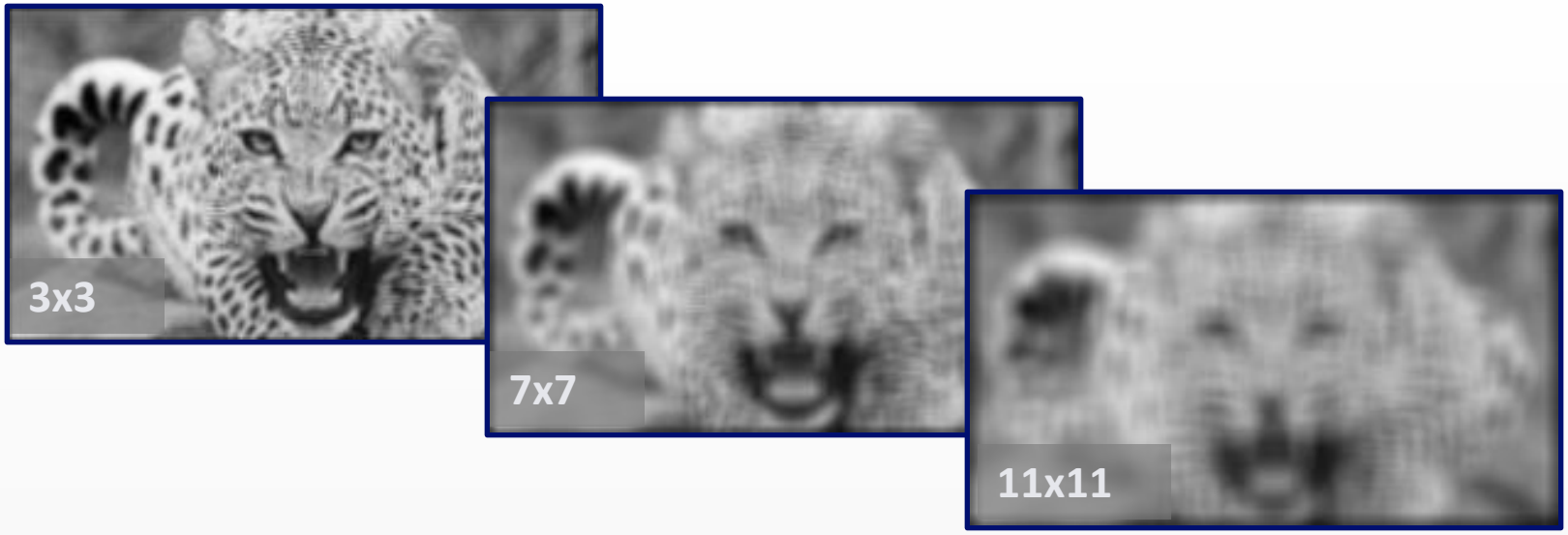
E.g.: $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$ $\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix}$

  - 0: edge detection

E.g.: $\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$ $\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix}$

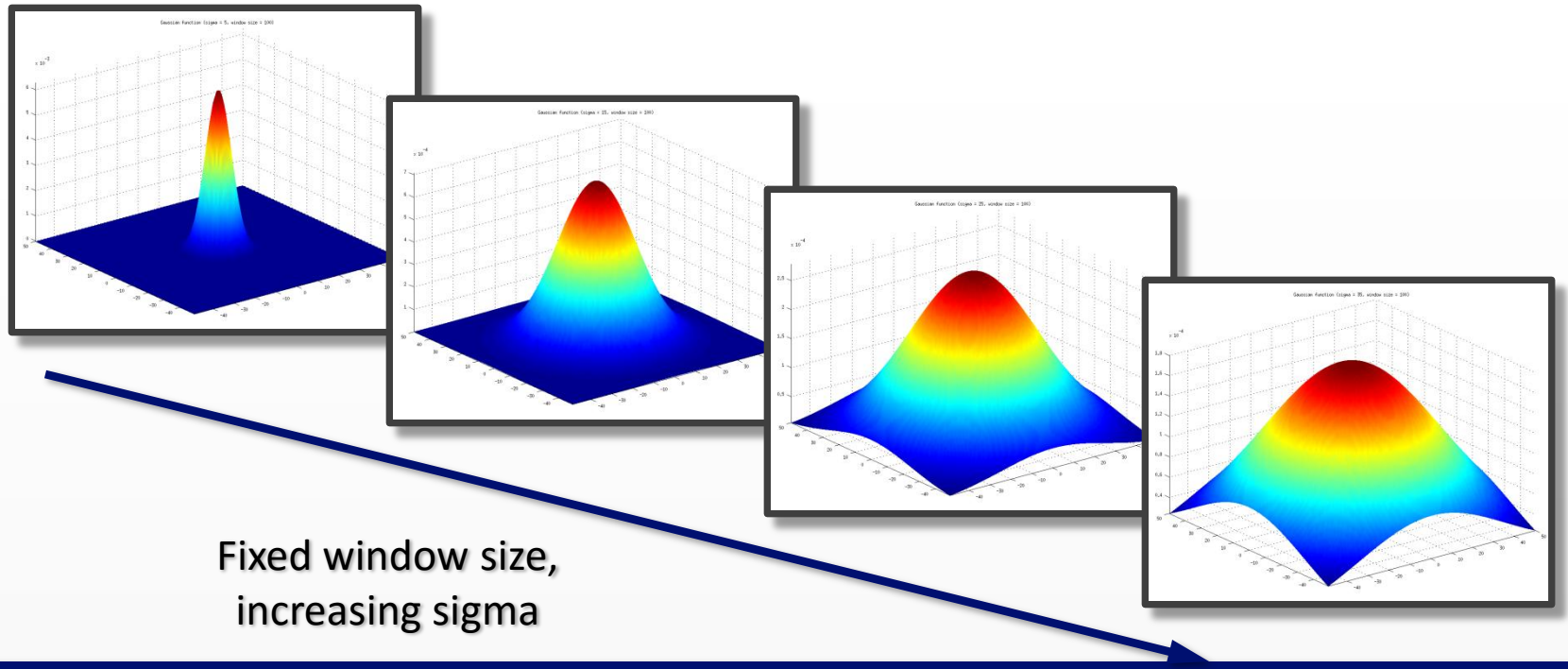
Smoothing/Blurring

- Simple average:



Smoothing/Blurring

- Gaussian blur:
  - Weights are defined by a 2D Gaussian function
  - 2 parameters: **size of the window** and the **standard deviation** of the Gaussian



Smoothing/Blurring

- Gaussian blur:
  - Weights are defined by a 2D Gaussian function
  - 2 parameters: window size and the width of the Gaussian
  - E.g. kernel size = 5x5; σ = 1.5;

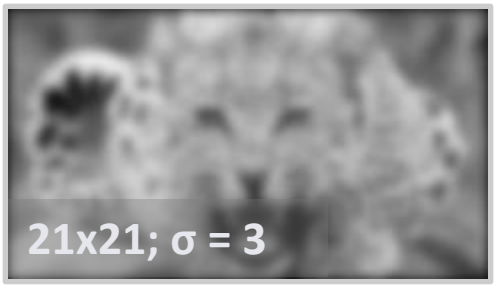
$$\begin{bmatrix} 0.0144 & 0.0281 & 0.0351 & 0.0281 & 0.0144 \\ 0.0281 & 0.0547 & 0.0683 & 0.0547 & 0.0281 \\ 0.0351 & 0.0683 & 0.0853 & 0.0683 & 0.0351 \\ 0.0281 & 0.0547 & 0.0683 & 0.0547 & 0.0281 \\ 0.0144 & 0.0281 & 0.0351 & 0.0281 & 0.0144 \end{bmatrix}$$

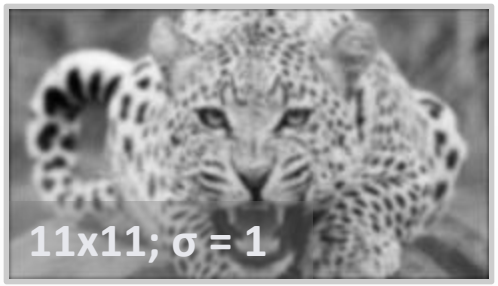
  - E.g. kernel size = 3x3; σ = 1.5;

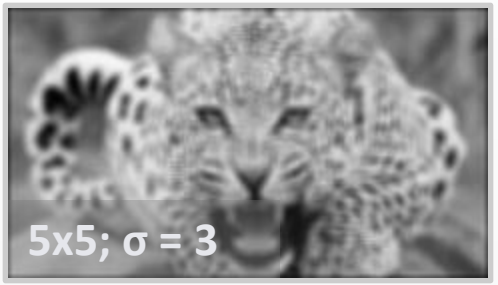
$$\begin{bmatrix} 0.0947 & 0.1183 & 0.0947 \\ 0.1183 & 0.1478 & 0.1183 \\ 0.0947 & 0.1183 & 0.0947 \end{bmatrix}$$

Smoothing/Blurring

- Gaussian blur:







Edge Detection

- ***Edges:*** locations on the image where the ***intensity changes sharply*** (usually at the contour of objects)

- We are searching for places where the gradient of the 2D function (the image) is high.
- Main types of edge detection:
  - First order derivative
  - Second order derivative
  - Others:
    - Complex methods e.g. Canny method
    - Phase Congruancy

Edge Detection

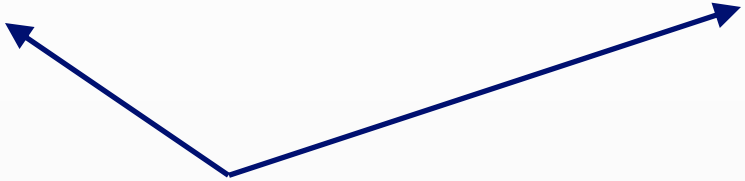
- **Edge detection with first order derivative:**
  - Using the gradient vector:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^T$$

  - The approximation of the partial derivatives:

$$\frac{\partial f}{\partial x} = \lim \frac{f(x+dx, y) - f(x, y)}{dx} \qquad \frac{\partial f}{\partial y} = \lim \frac{f(x, y+dy) - f(x, y)}{dy}$$

$$\approx f(x+1, y) - f(x, y) \qquad \approx f(x, y+1) - f(x, y)$$



Since the smallest meaningful discrete value is *dx*=1 and *dy* =1.

Edge Detection

- **Edge detection with first order derivative:**
  - The approximation of the partial derivatives:

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x, y) \qquad \frac{\partial f}{\partial y} \approx f(x, y+1) - f(x, y)$$

  - Kernels for the gradient calculation with convolution (Prewitt):

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

For better localization

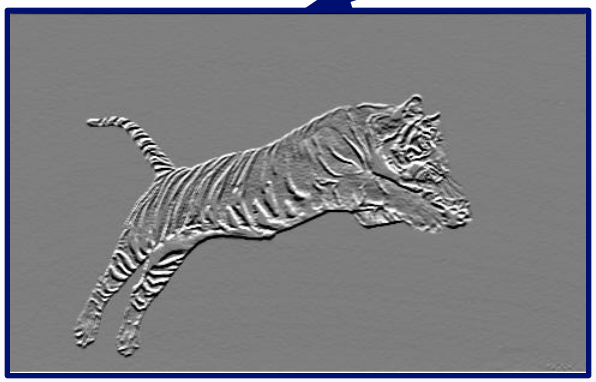
For noise reduction

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

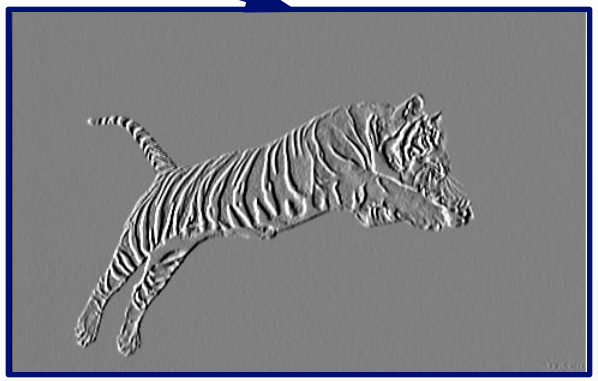
Edge Detection

- **Edge detection with first order derivative:**

Prewitt detector



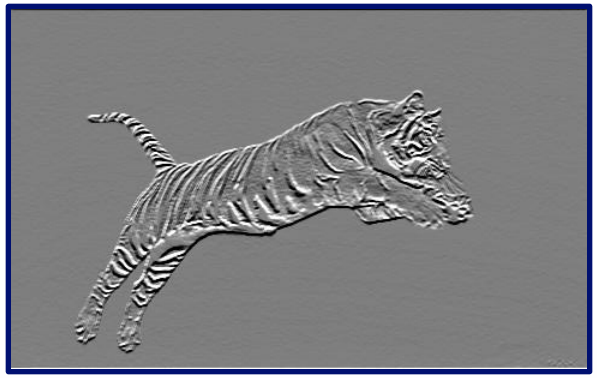
Horizontal gradient image



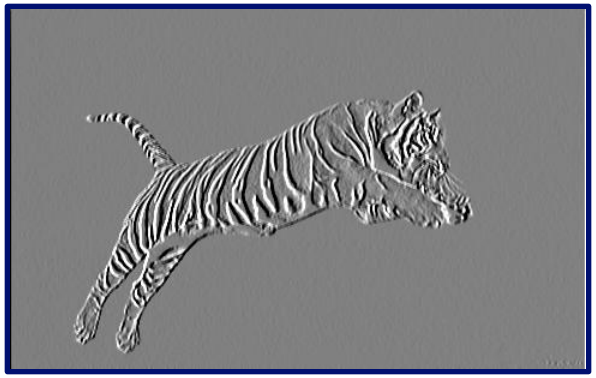
Vertical gradient image

Edge Detection

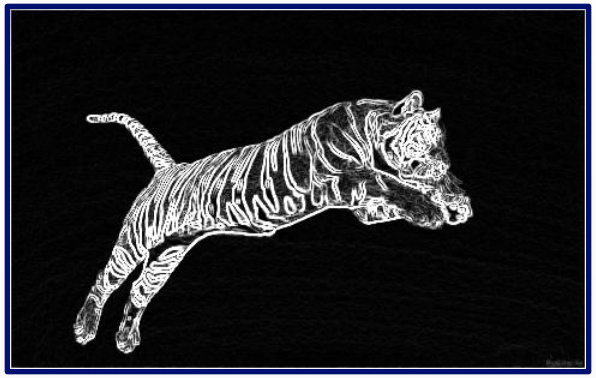
- **Edge detection with first order derivative:**



Horizontal gradient image



Vertical gradient image



gradient image

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Edge Detection

- **Edge detection with second order derivative:**
  - Approximation of the second order derivative for *x* direction:

$$\frac{\partial^2 f}{\partial x^2} = \lim_{d \to 0} \frac{\frac{\partial f}{\partial x}(x+d) - \frac{\partial f}{\partial x}(x)}{d} \approx \frac{\partial f}{\partial x}(x+1) - \frac{\partial f}{\partial x}(x) =$$

$$= f(x+2) - 2f(x+1) + f(x)$$

(*y* direction similarly)

  - Kernel for the second order gradient calculation with convolution:
    - Laplace operator:

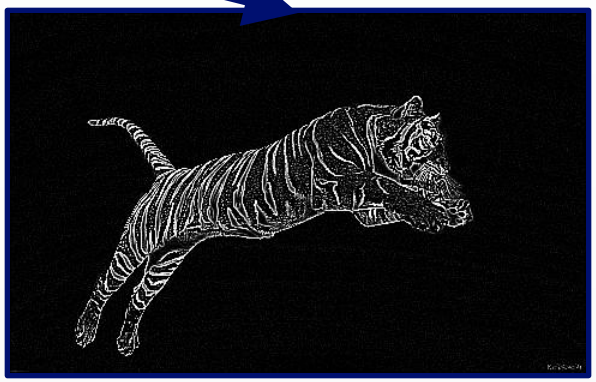
$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

    - There are other variations. (e.g. Second order Prewitt)

Edge Detection

- **Edge detection with second order derivative:**

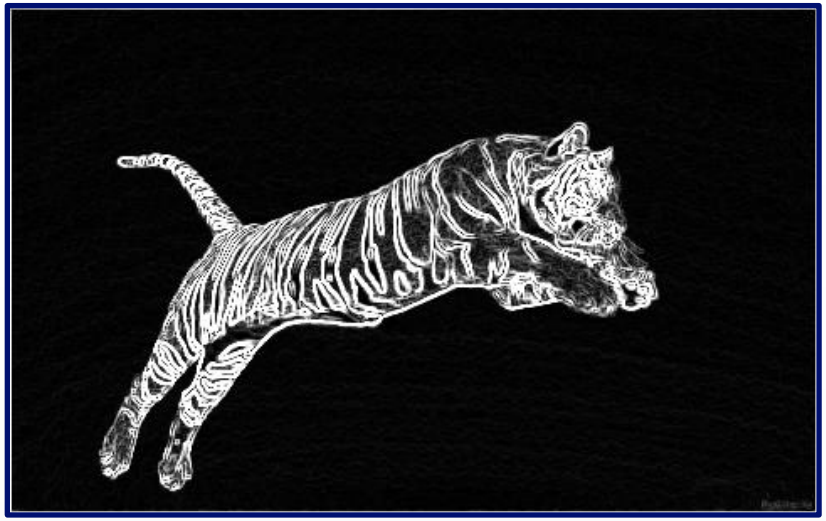
Laplace edge detector



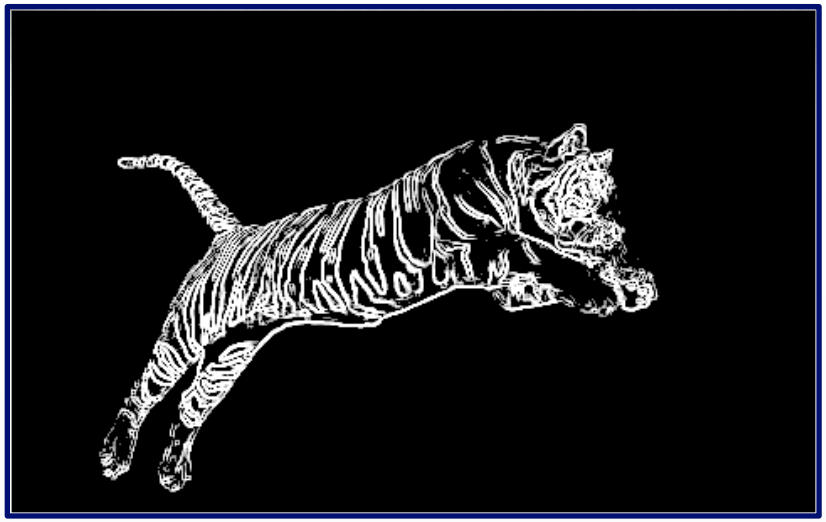
Prewitt 2nd order detector

Edge Detection

- **Thresholding**:
  - To eliminate weak edges, a threshold can be used on the gradient image:



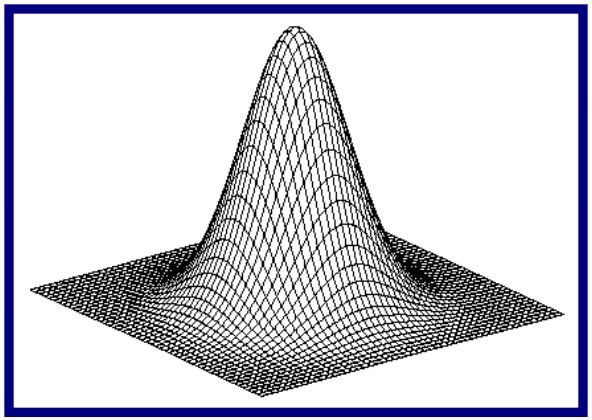
Prewitt first order gradient image



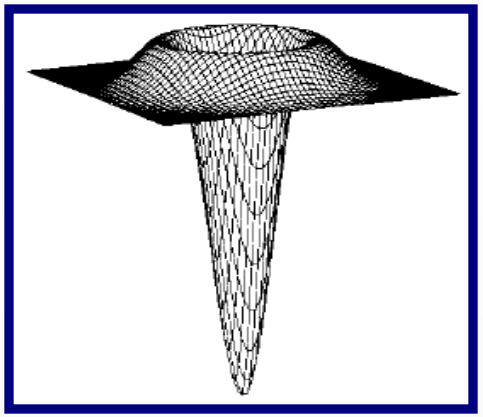
Prewitt first order gradient image with threshold = 120

Reducing the effect of noise on edege images

- Edge detection with noise reduction:
  - 1. step: Noise reduction by convolution with Gaussian filter
  - 2. step: Edge detection by convolution with Laplacian kernel
- Since convolution operation is associative we can convolve the Gaussian smoothing filter with the Laplacian filter first, and then convolve this hybrid filter (***Laplacian of Gaussian: LoG***) with the image.



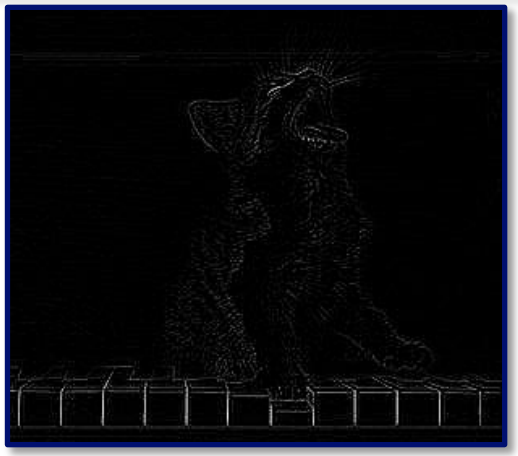
Gaussian function



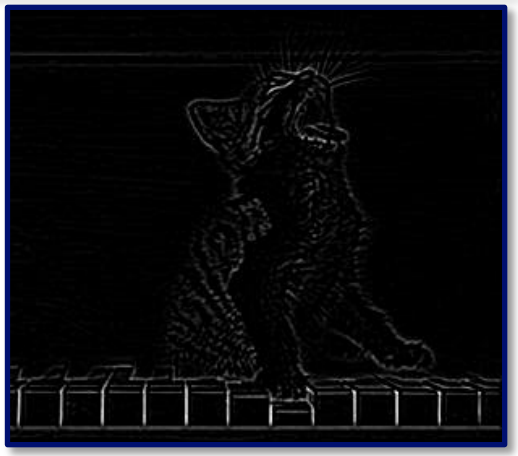
Laplacian of Gaussian

Laplacian of Gaussian

Original Image

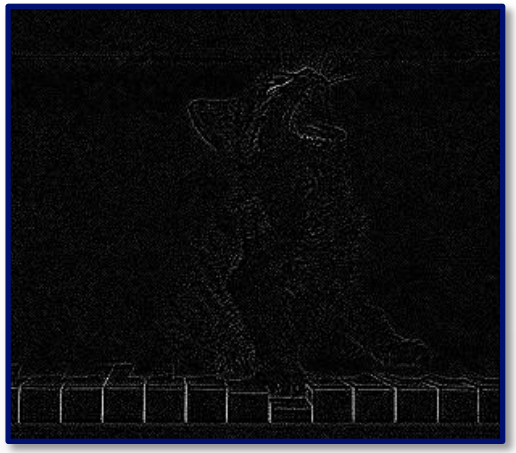


Laplacian

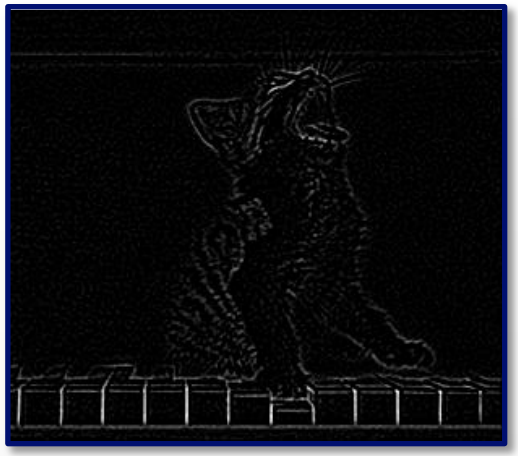


Laplacian of Gaussian

Noisy Image



Laplacian



Laplacian of Gaussian

Edge Enhancement

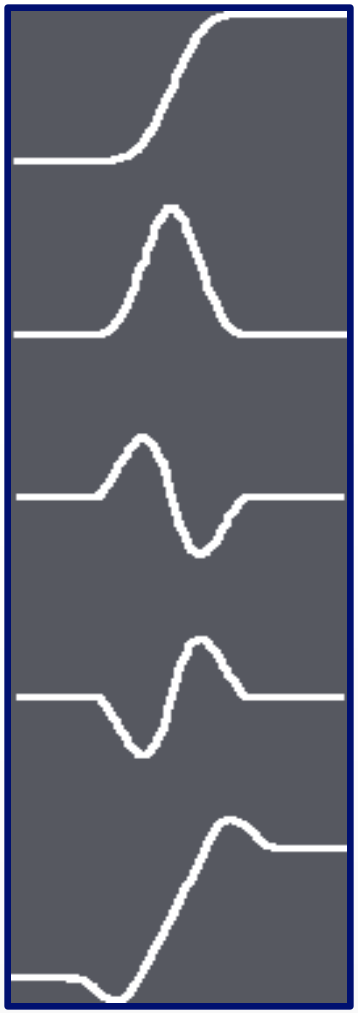
$f(x)$

$\frac{\partial f}{\partial x}$

$\frac{\partial^2 f}{\partial x^2}$

$-\frac{\partial^2 f}{\partial x^2}$

$f(x) - \frac{\partial^2 f}{\partial x^2}$



Kernel for edge enhancement with Laplace operator:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Original image

Edge enhanced image

Canny Edge detector

- Properties of a „good” edge detector:
  - Good detection:
    - detects as many real edges as possible
    - does not create false edges (because of e.g. image noise)
  - Good localization:
    - the detected edges should be as close to the real edges as possible
  - Isotropic:
    - all edges are detected regardless of their direction

- John F. ***Canny*** has developed an edge detector in 1986 to meet these requirements.

Canny Edge detector

- Main steps of the algorithm:
  1. ***Noise reduction:***
     - The original image is convolved with a ***Gaussian kernel*** to reduce image noise.
  2. ***Gradient intensity and direction calculation:***
     - The horizontal and vertical derivative image is calculated (e.g. with Prewitt kernel)
     - Based on them the gradient intensity and direction can be calculated:

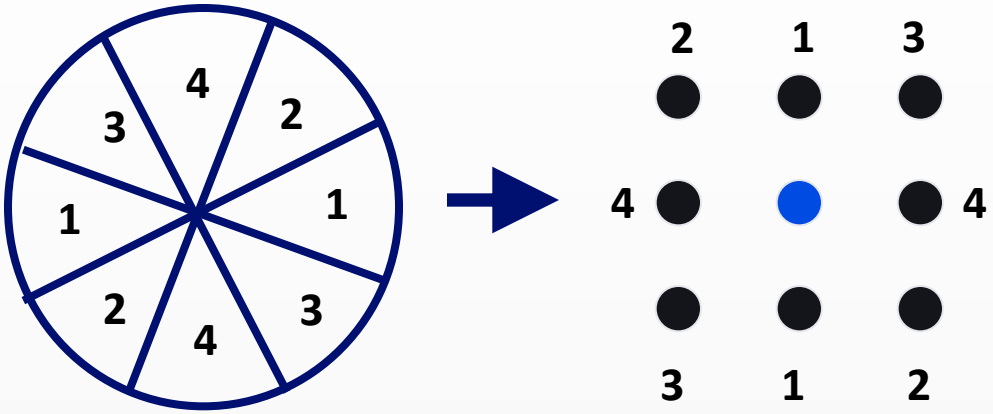
$$d^x = \left(\frac{\partial f}{\partial x}\right) \qquad d = \|\nabla f\| = \sqrt{\left(d^x\right)^2 + \left(d^y\right)^2}$$

$$d^y = \left(\frac{\partial f}{\partial y}\right) \qquad \Theta = \arctan\left(\frac{d^x}{d^y}\right)$$

Canny Edge detector

3. Non-Maximum Suppression:

- The goal is thinning the edges.
- Each edge is categorized into one of *4 main edge directions* (0°, 45°, 90°, 135°), based on the gradient direction image (θ).
- At every pixel, it suppresses the edge, by setting its value to 0, if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction:



Canny Edge detector

4. ***Hysteresis thresholding:***
   - Problem with simple thresholding:
     - if the threshold is low, many false edges will appear
     - if the threshold is high, true edges will disappear
   - Solution: using two threshold instead of only one: t_1, t_2, where $t_1 > t_2$
     - if the edge magnitude at *(i,j)* point is higher than t_1 then it is an edge
     - if the edge magnitude at *(i,j)* point is lower than t_2 then it is not an edge
     - if the edge magnitude at *(i,j)* point is lower than t_1 but higher than t_2, then it is an edge, only if one of its neighbors in the direction of $\vartheta(i,j)$ is an edge.

original image

gradient intensity

non-maximum suppression

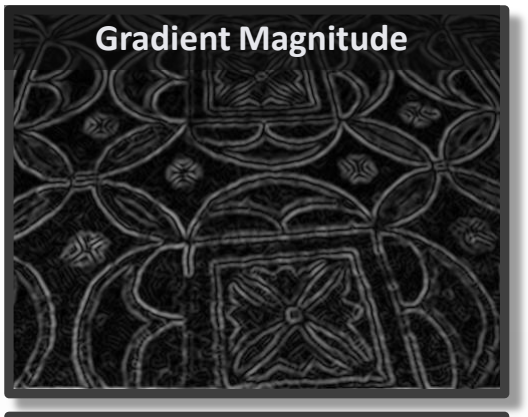
hysteresis thresholding

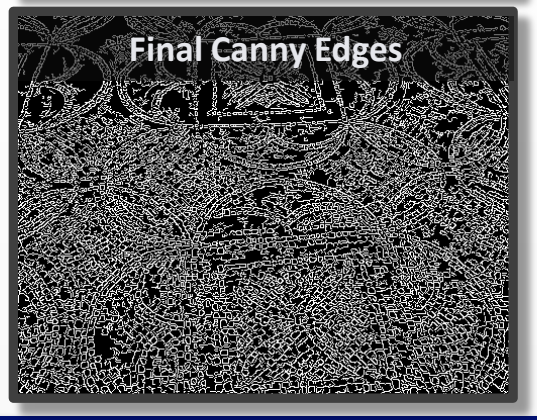

Original Image


Weakly Blurred Image


Moderately Blurred Image


Gradient Magnitude

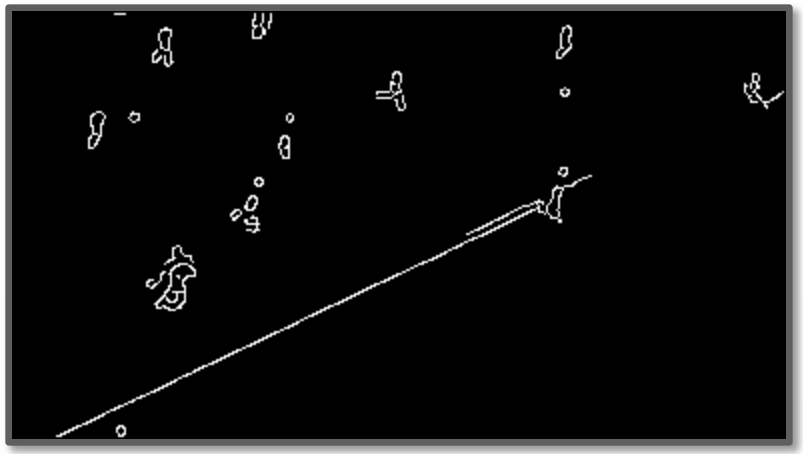

Gradient Magnitude


Final Canny Edges


Final Canny Edges

Hough Transformation

- Another 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.

Hough Transformation

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

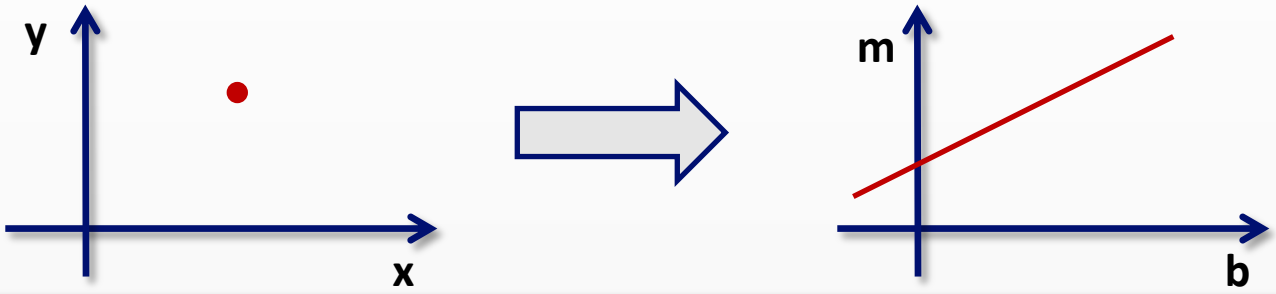
$$y = mx + b$$

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

  - The above equation can be re-writtten in terms of ***m*** and ***b:***

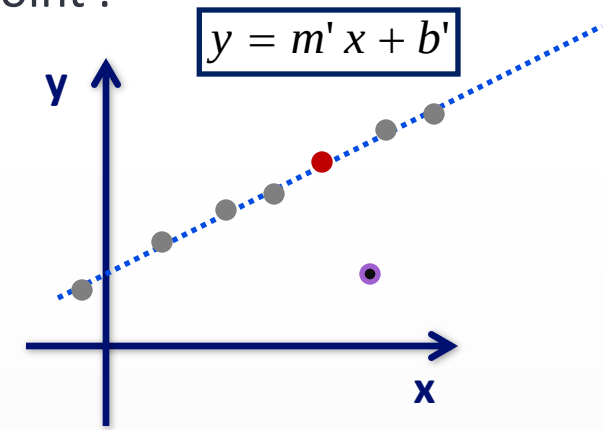
$$m = -\frac{1}{x}b + \frac{y}{x}$$

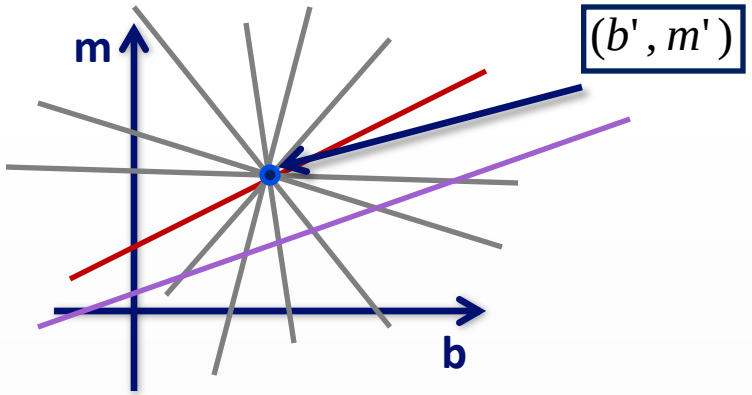
  - For a fixed ***y = y', x = x'*** point in the image space, we get a line in the **(m, b) space** with a slope ***-1/x'*** and an m-intercept: ***y'/x'***:



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 :

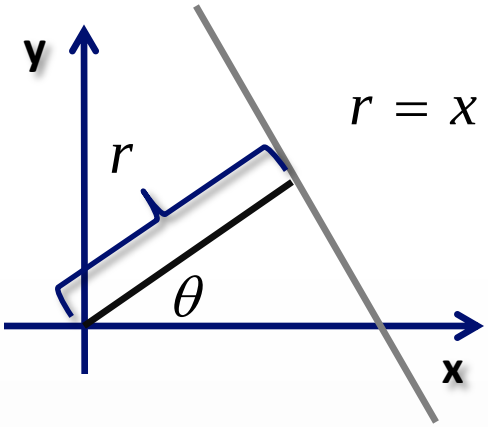




  - This point will be ***m=m'*** and ***b=b'***, the slope and intercept of the line in the image space. ⇨ We have the equation of the line!
- But, there is a problem with this equation of the line: vertical lines cannot be described (their slope would be infinite).

Hough Transformation

- To be able to describe all possible lines we will use the polar equation of the line:

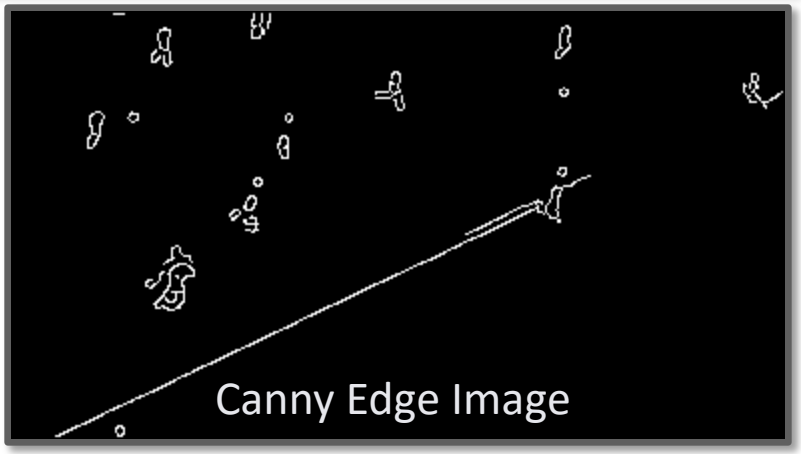


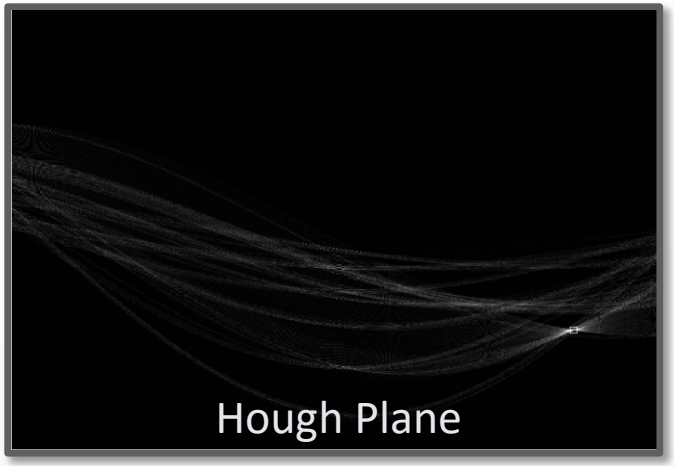
$$r = x \cdot \cos\theta + y \cdot \sin\theta$$

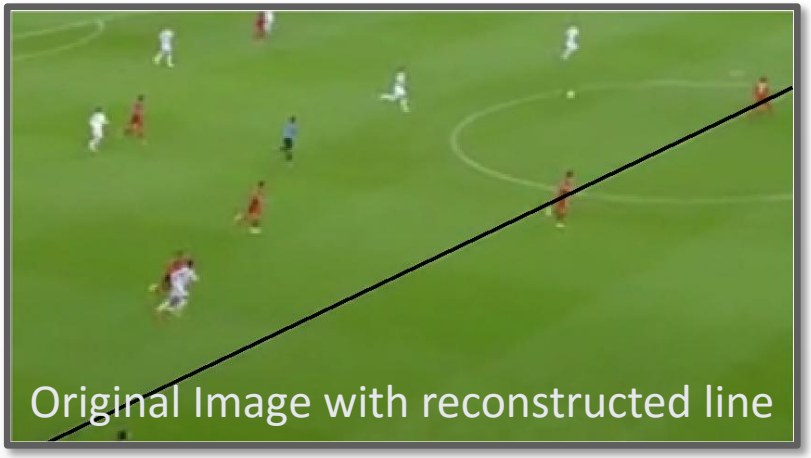
- The (r, θ) parameter space is called Hough space.
- A point in the Euclidian space is a sinusoid in the Hough space, described by the following equation: $r(\theta) = x \cdot \cos\theta + y \cdot \sin\theta$

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

Hough Transformation







Sources

Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

Wikipedia: http://en.wikipedia.org/wiki/Canny_edge_detector#Non-maximum_suppression

Image Processing slides of Csaba Benedek: „Konvolúció, élkereső eljárások. Fotometriai alapok.” 2008