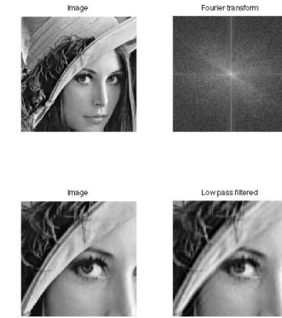


wikipedia.org



Computer Vision

Image Formation and Image Processing Basics



© wikipedia

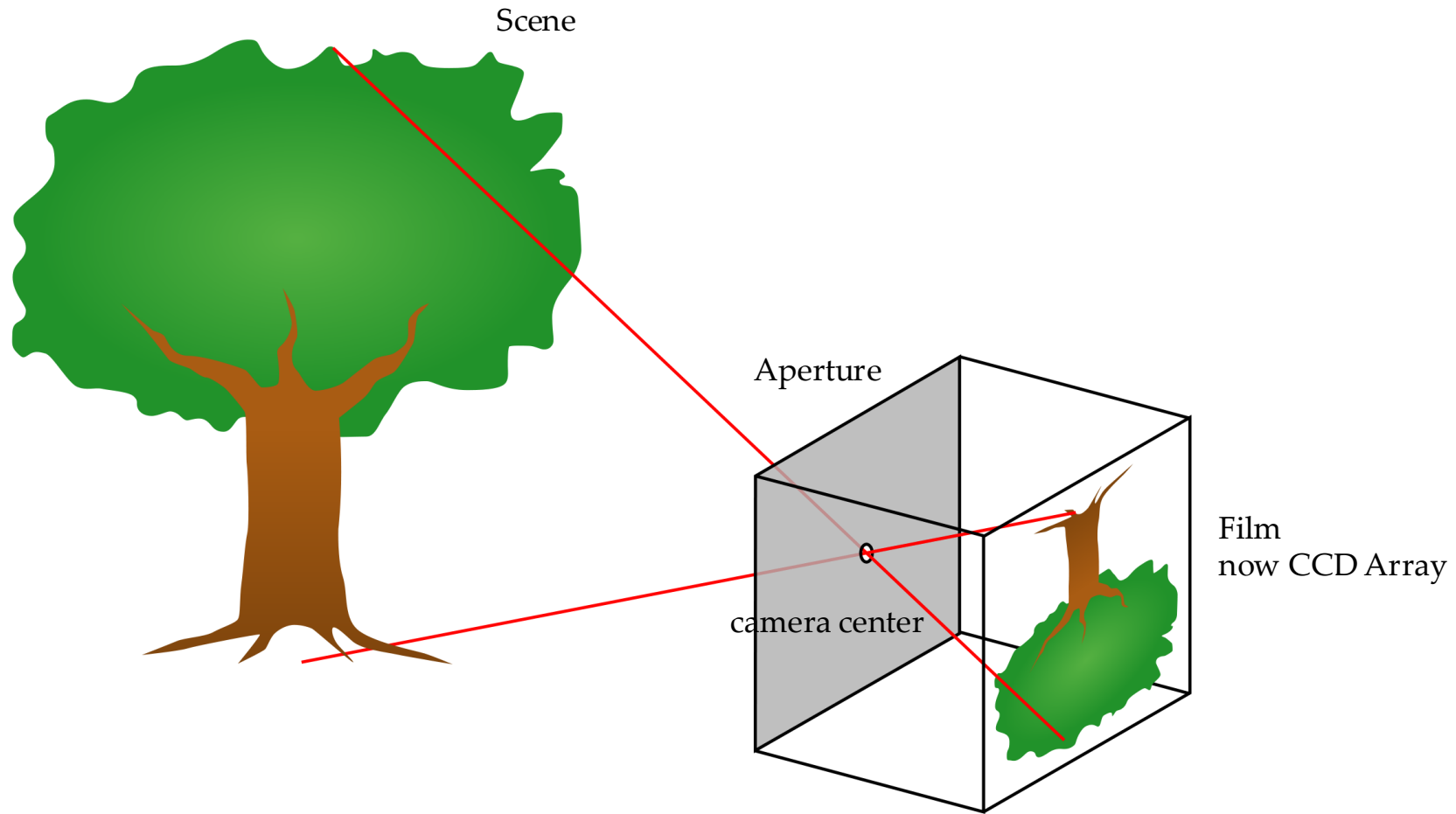
Master DataScience
Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de



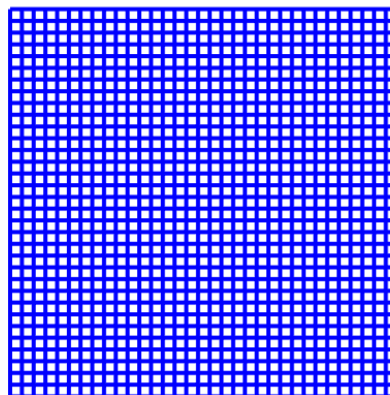
Goals for today

- Image formation (intrinsic / extrinsic camera parameters)
- Image representation
- Image processing
 - Filter (Convolution)
 - Gradients
 - Algorithms, e.g. Canny-Edge, Color Quantization
- *Assignment 1 is out*

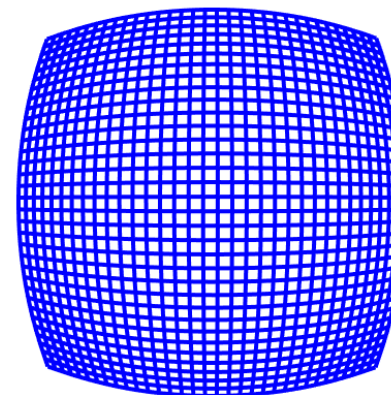
Image formation



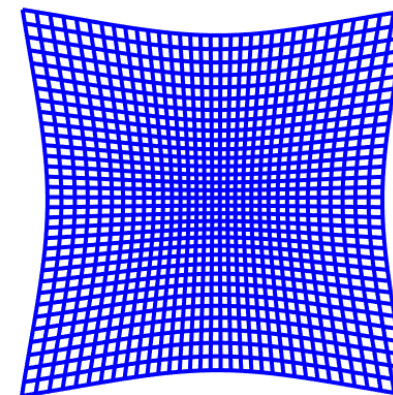
wikipedia.org



INPUT GRID



BARREL DISTORTION

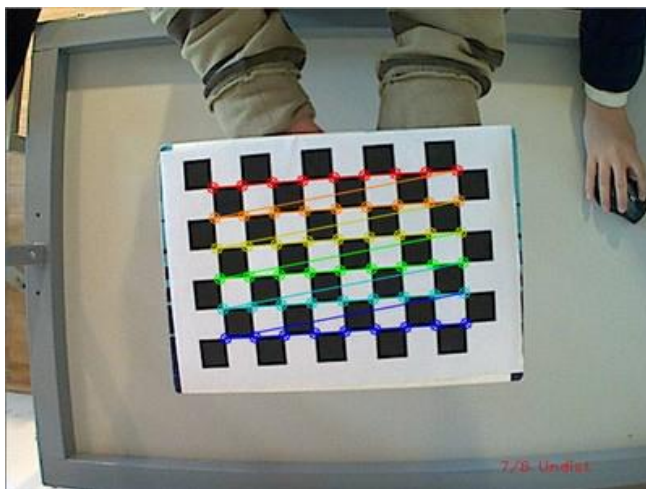


PINCUSHION DISTORTION

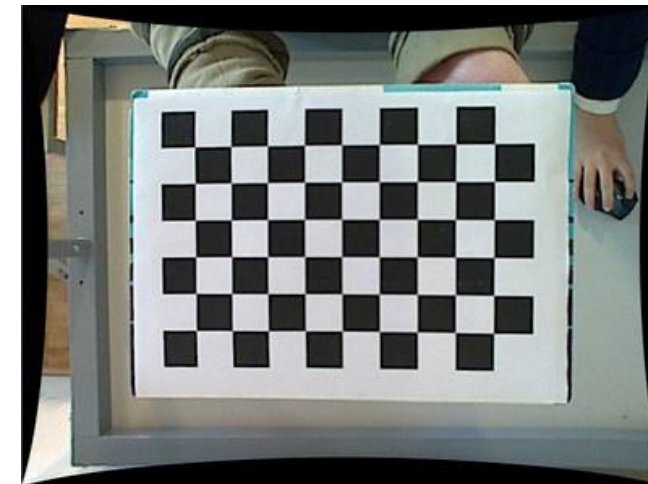
$$x_{distorted} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{distorted} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html



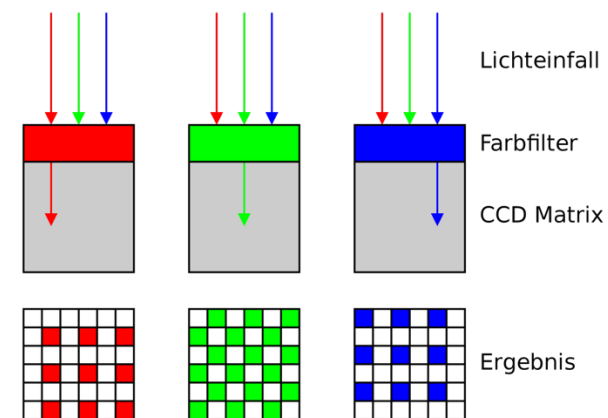
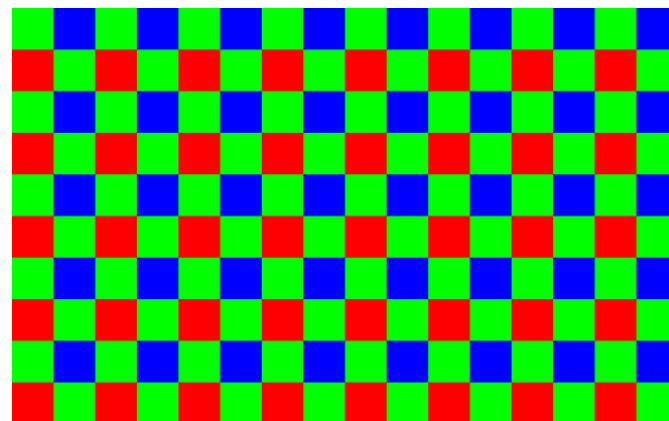
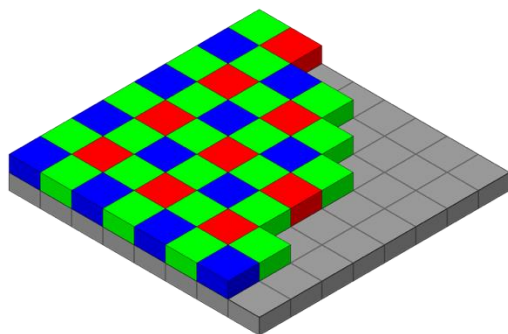
Undistort
e.g. with OpenCV



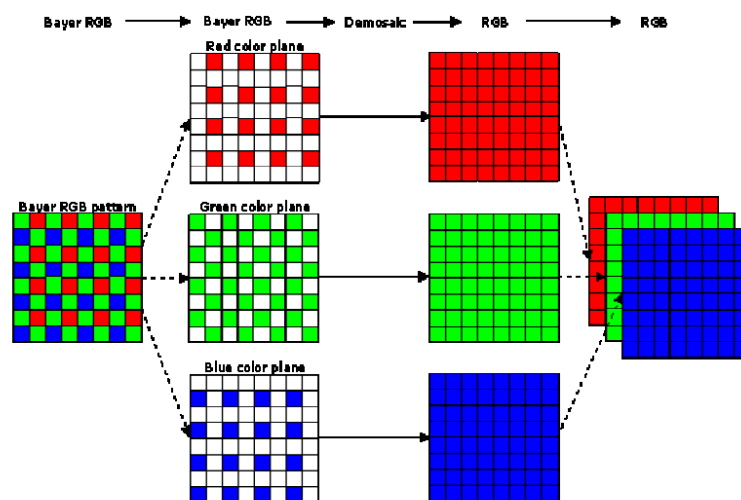
https://www.youtube.com/watch?v=ViPN810E0SU&ab_channel=OpenCVTutorials

Intrinsic and Extrinsic Camera Parameters

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K(R \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t)$$



<https://de.wikipedia.org/wiki/Bayer-Sensor>

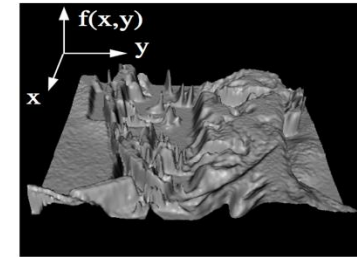
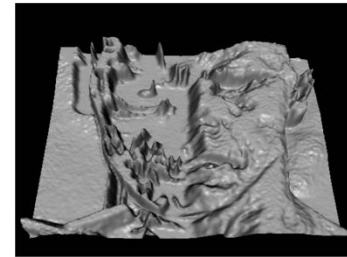
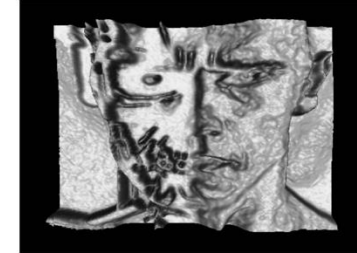


<http://dias.library.tuc.gr/view/manf/25161>

Image representation

Images are functions

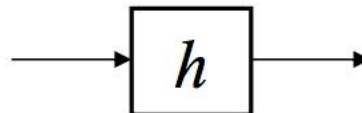
- **convex, continuous, differentiable**
- $I(x,y)$ are intensities at location (x,y)
- change of signal through transformation function h



N. Snavely



- $g(x) = h(f(x))$



Invert, Histogram,
Contrast
etc.

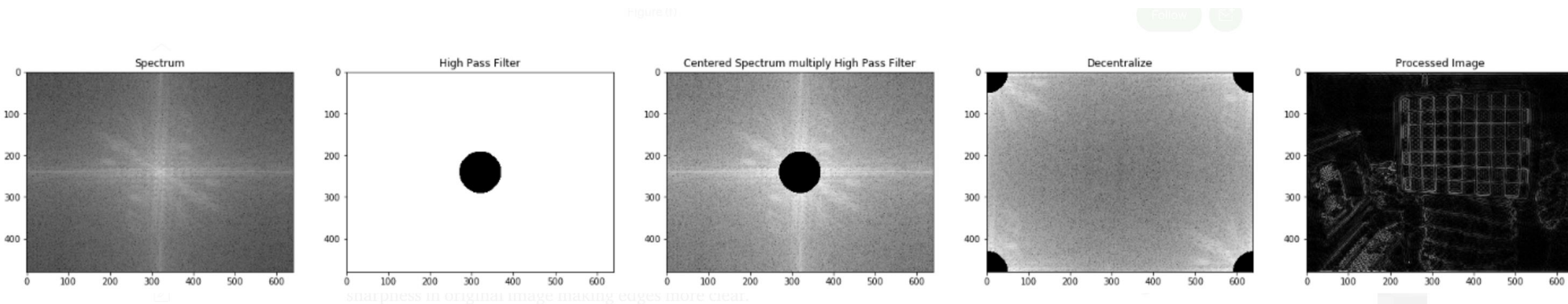
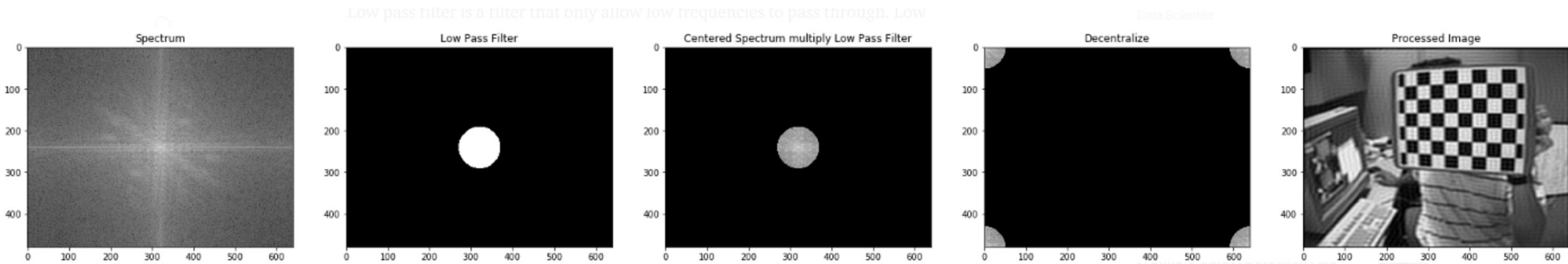
Fredo Durand, MIT

Fourier Transform

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) e^{-i2\pi(\frac{ki}{N} + \frac{lj}{N})}$$

$$e^{ix} = \cos x + i \sin x$$

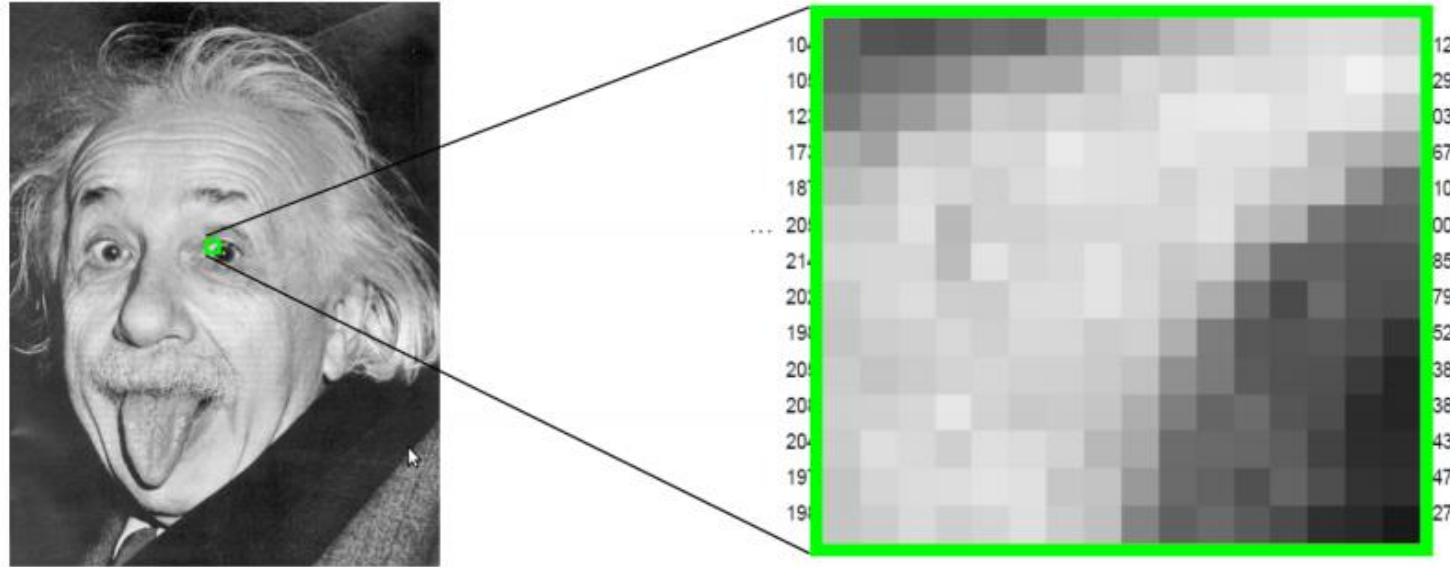
<http://www.jezzamon.com/fourier/index.html>



sharpness in original image making edges more clear.

Error Diffusion Dithering—Stucki with Python

Image representation



D. Schlesinger, TU Dresden

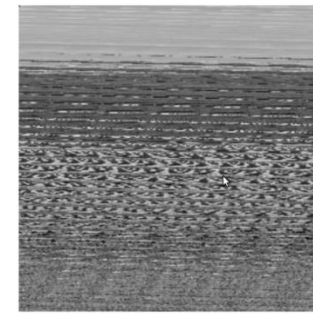
Matrices

or vectors

$$A_{n \times m} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

↑
Pixel's intensity value

Images are Matrices



D. Schlesinger, TU Dresden

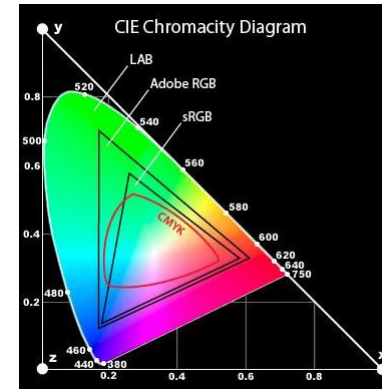
- similar vectors are not necessarily similar images
- similar images are not necessarily similar vectors
- Representation, geometric or color transformation (RGB vs. LAB color space)



RGB – Color space



LAB – Color space



Youtube Video in Moodle
that explains color spaces

<https://www.photo.net/learn/using-lab-color-adjustments/>

Compute distances between colors in LAB (differences correspond more naturally to human color perception)

Images as graphs

- Pixel are nodes with 4- (or 8-) neighbors
- Edges are weights
 - e.g. color distances between pixels
- Graph is a grid
 - nodes can be labeled for segmentation etc.
- **Algorithms:**
 - Graph Cut
 - Conditional Random Fields (CRF)

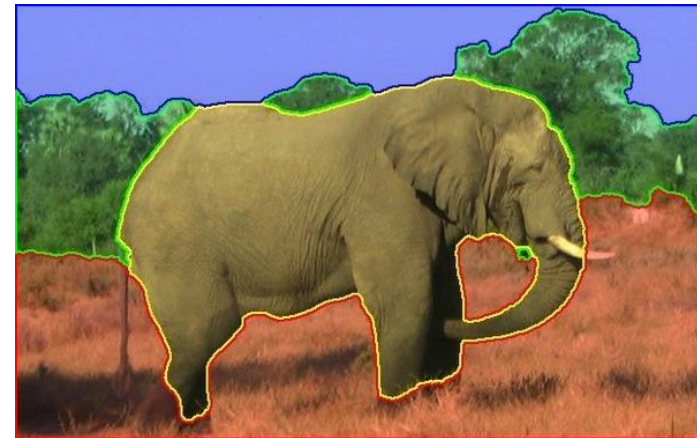
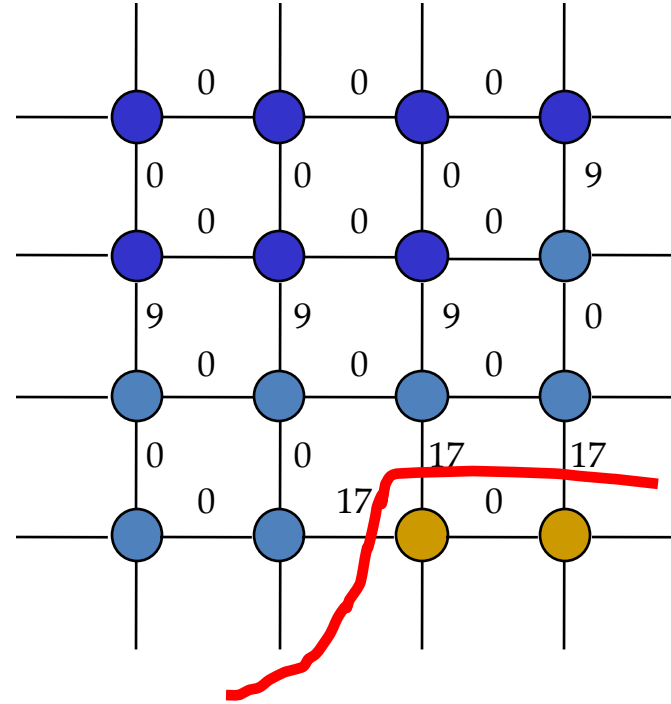
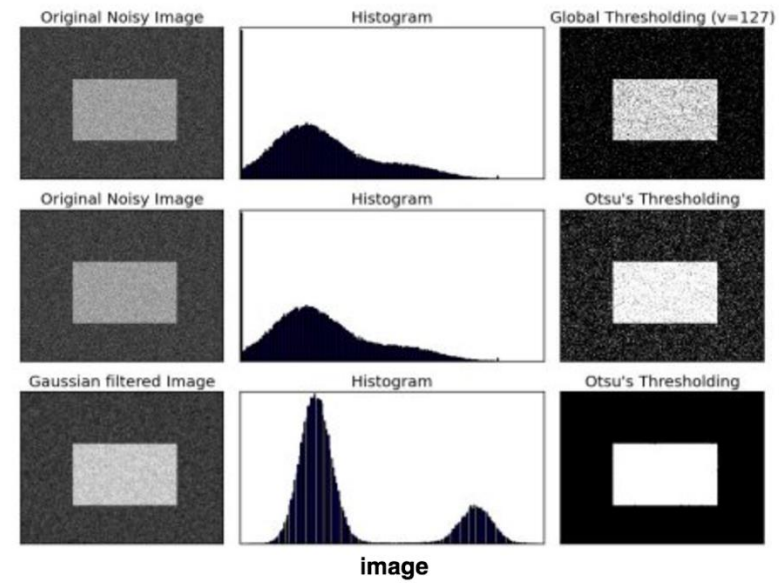
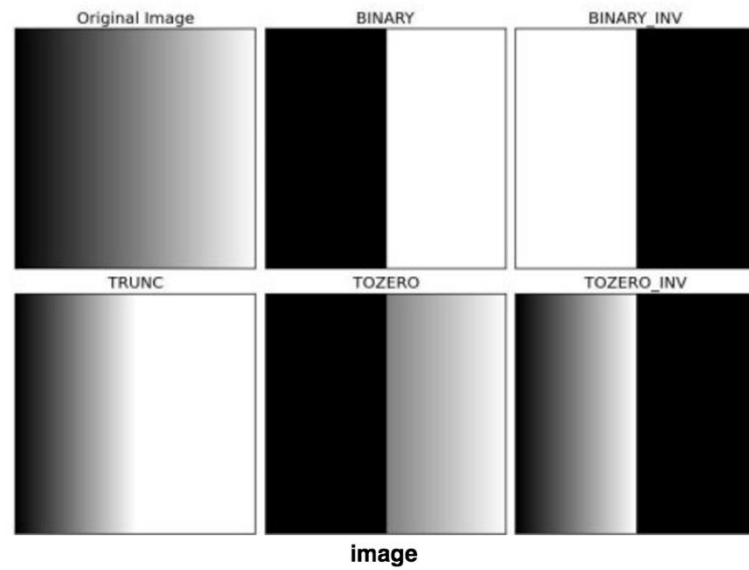


Image Processing

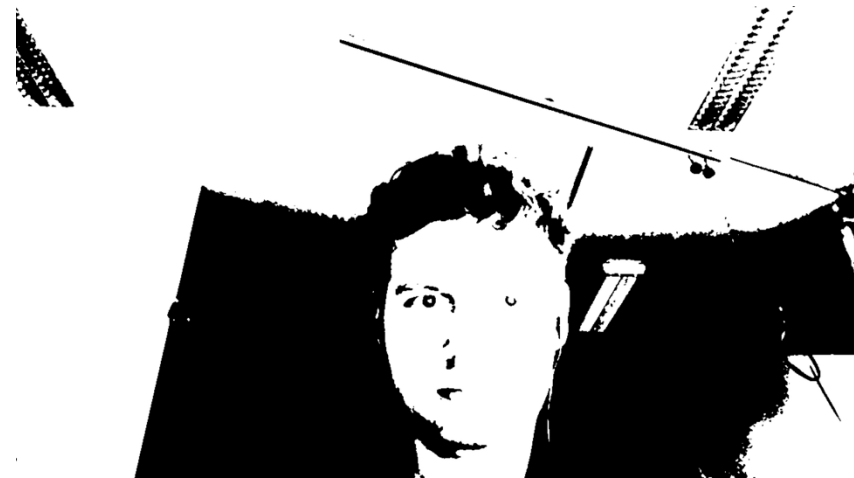
What is OpenCV?

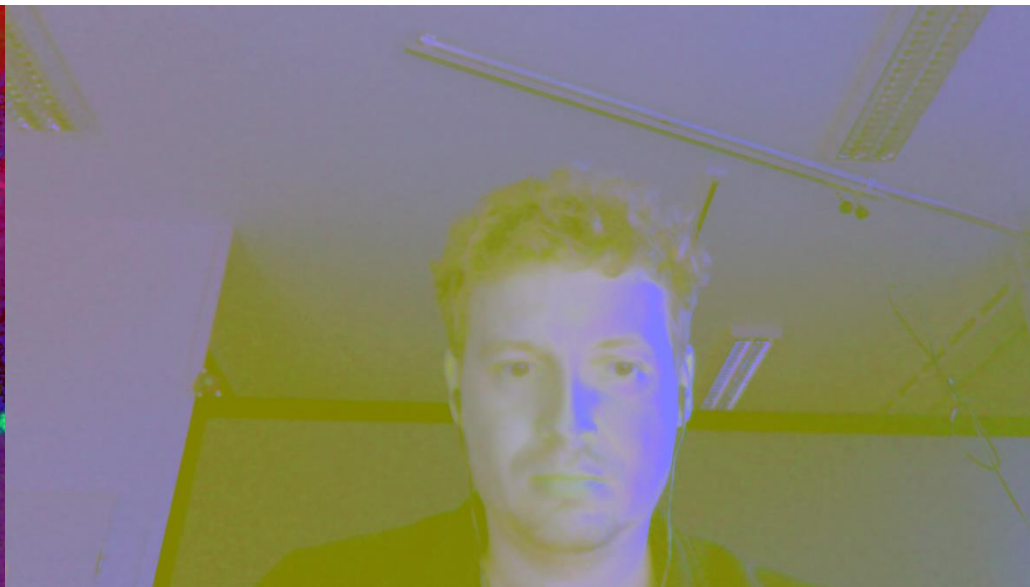
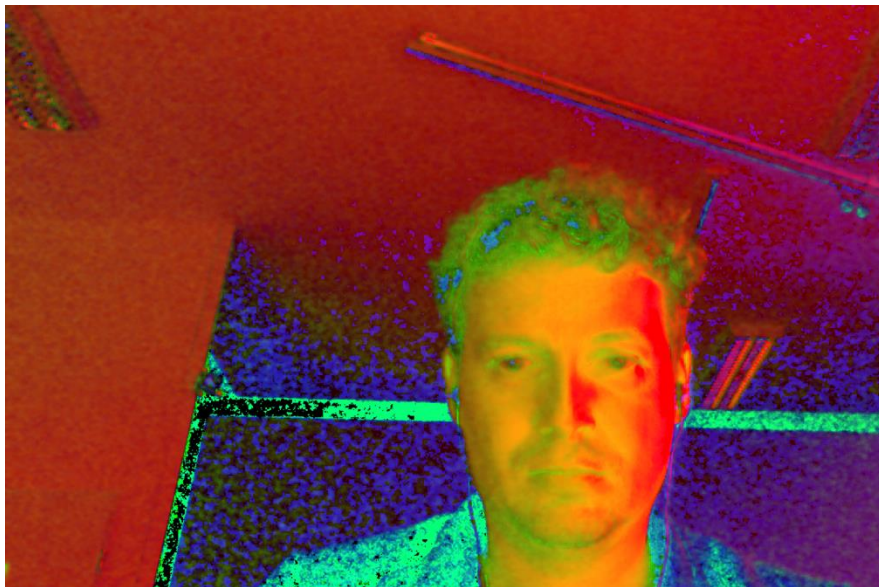
Image processing

- Thresholding
- Changing Colorspaces
- Convolution + Image Gradients
- Canny Edge Detection
- Color Quantization with k-means

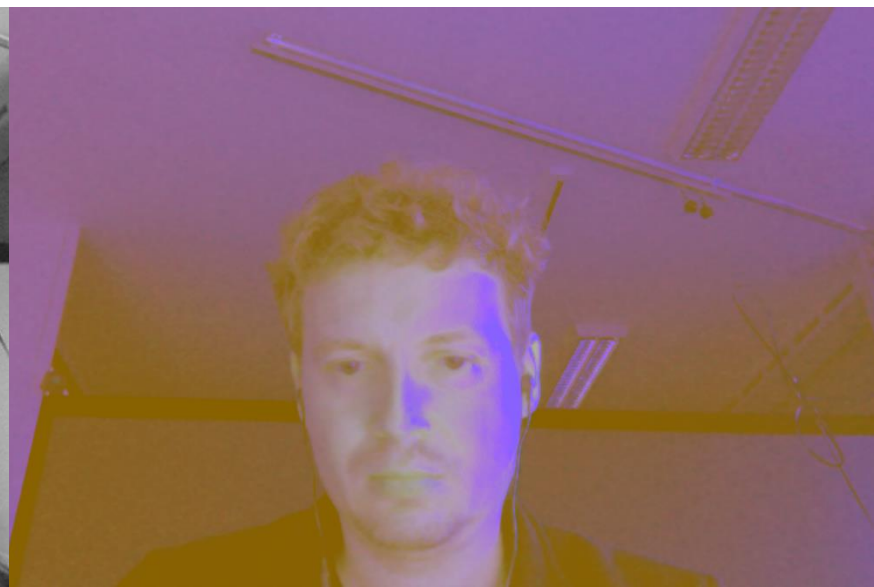


Thresholding





Colorspaces



Color Quantization



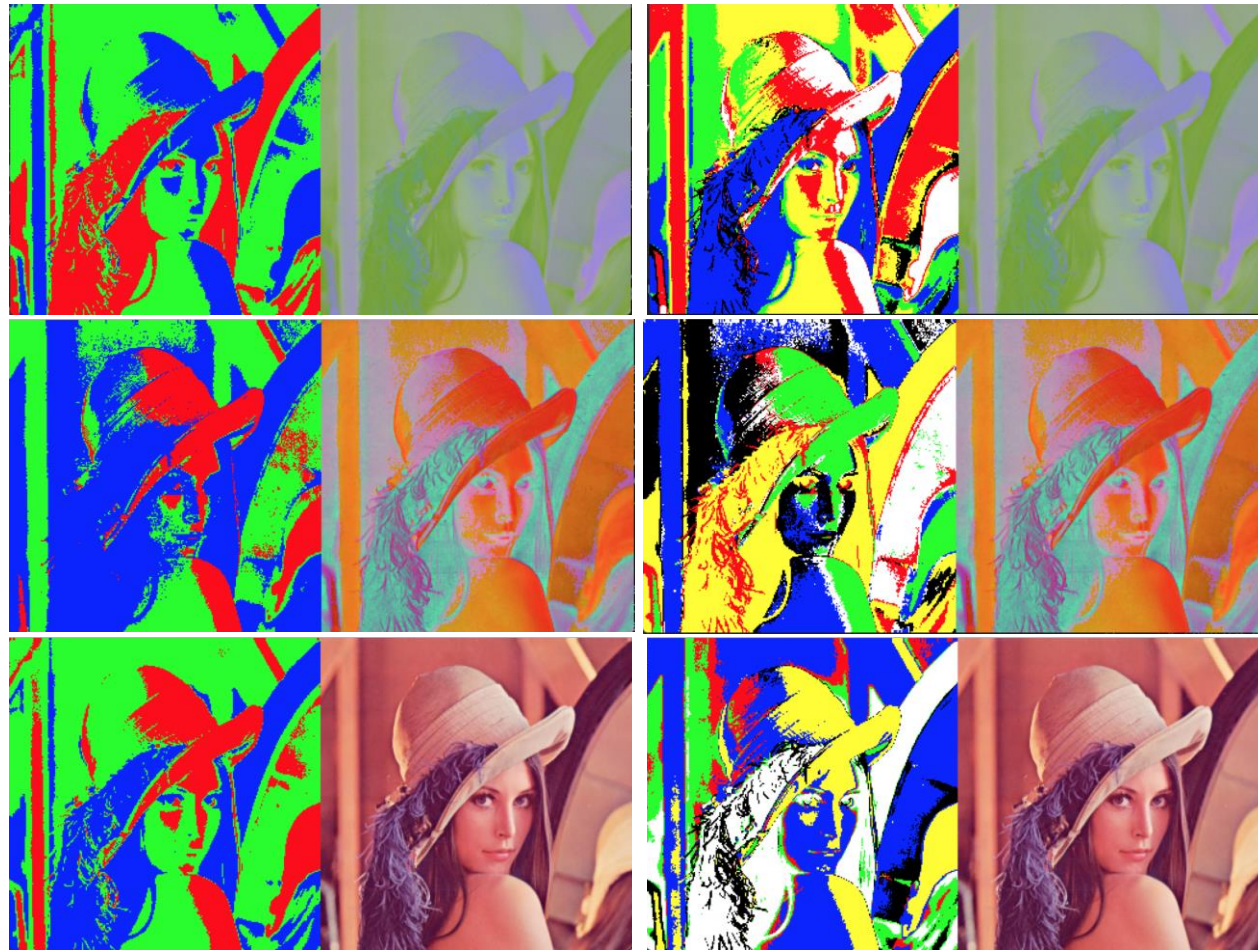
Original

k=4

k=16

k=32

k=64



(a)

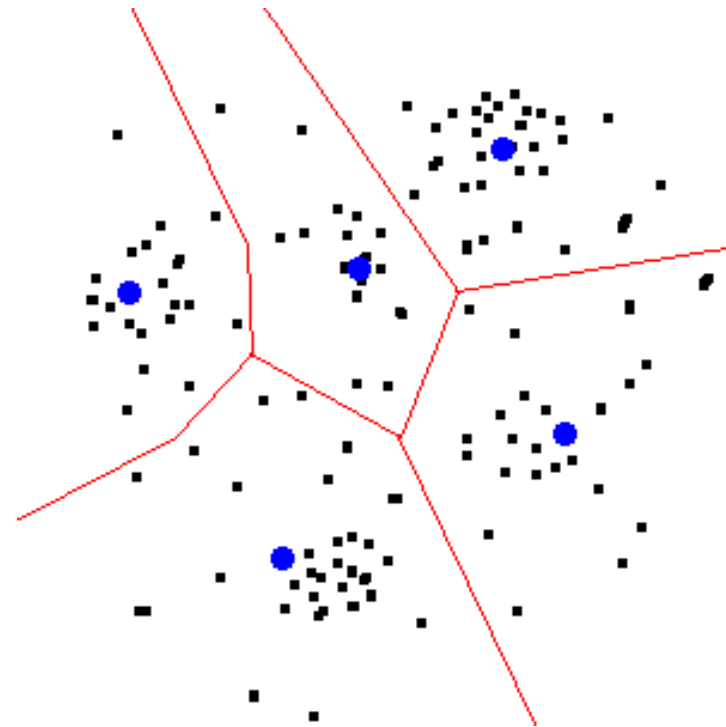
(b)

k-means

- important **clustering method** in CS
- partitions N data points in k clusters $S = \{S_1, S_2, \dots, S_k\}$
- each **data point belongs to cluster with nearest mean**
 - mean of the cluster is cluster center/
representative

$$\arg \min_s \sum_{i=1}^k \sum_{x \in s_i} \|x - \mu_i\|^2$$

- results in partitioning of data space (voronoi cells)

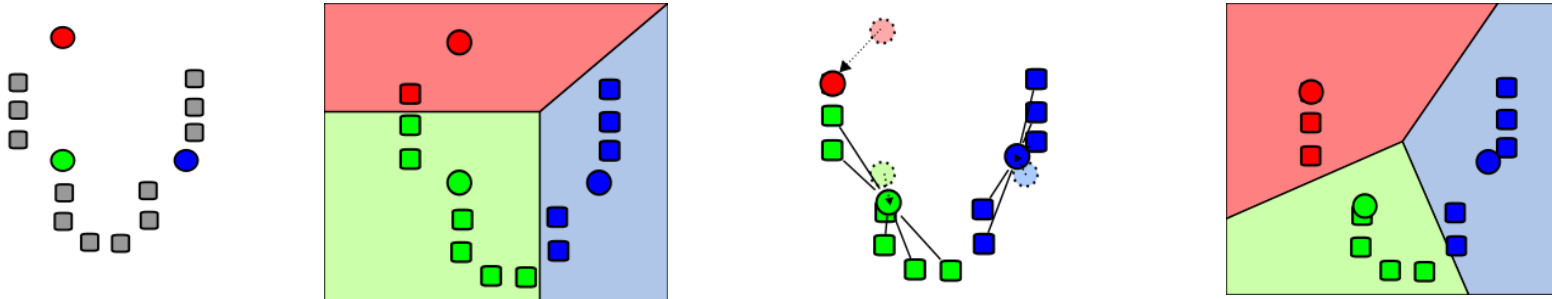


<http://mnemstudio.org/>

k-means Algorithm

Implement in homework 1

- **Steps:**
 1. **Initialization:** Randomly assign k data points as initial cluster centers (mean of the cluster)
 2. **Assignment step:** Assign each data point to the cluster with closest cluster center
 3. **Update step:** Calculate new centroids (mean points) of cluster
- **Iterate steps 2 /3** until convergence (no changes in centroid)



Convolution theorem

[https://subscription.packtpub.com/
book/big_data_and_business_intelligence/
9781789343731/3/ch03lvl1sec25/
convolution-theorem-and-frequency-domain-gaussian-blur](https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789343731/3/ch03lvl1sec25/convolution-theorem-and-frequency-domain-gaussian-blur)

$$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

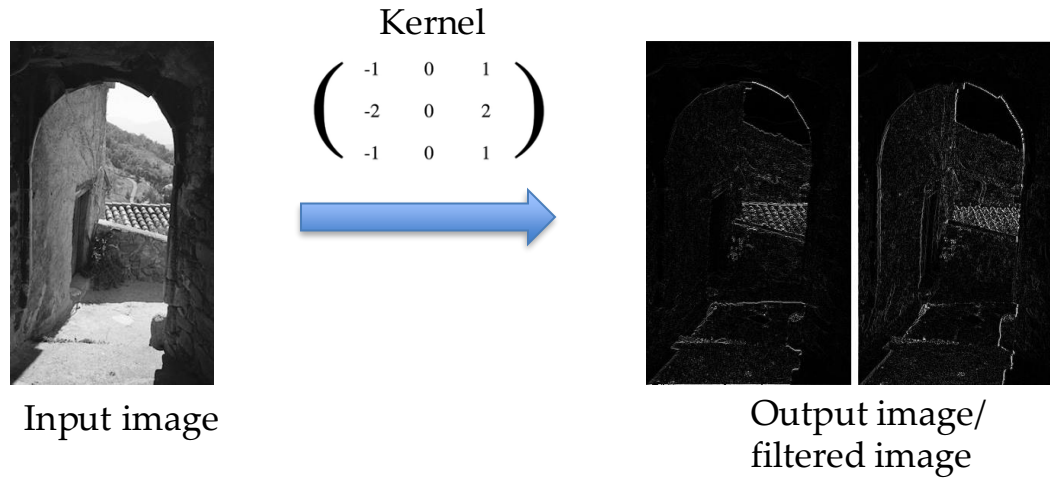
Space convolution = frequency multiplication

The convolution theorem says that **convolution** in an image domain is equivalent to a simple multiplication in the frequency domain

Convolution

Convolution

- filter frequency characteristics of the image
 - general approach for image effects/analysis
 - convolution done by structural element (kernel)

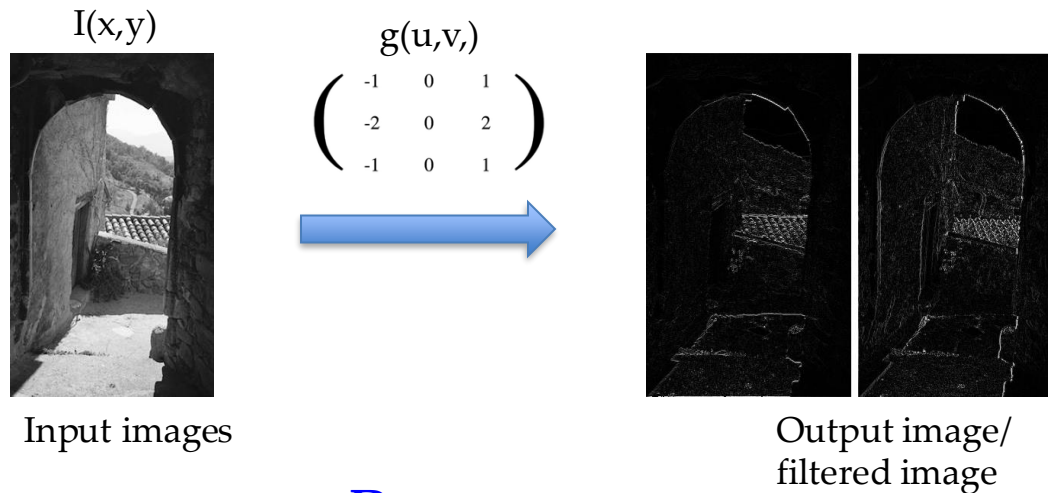


- Compute new intensity value of central pixel as weighted sum of its neighbors

Convolution

- Convolution is multiplication of pixel and its neighbors by kernel matrix

$$I' = \sum_{u,v} I(x - u, y - v)g(u, v)$$



[Demo](http://setosa.io/ev/image-kernels/)

<http://setosa.io/ev/image-kernels/>

What happens at borders?

- padding
- zero

Code yourself in first assignment

Image gradient - Sobel Filter

- Generation of an image gradient

$$\mathbf{G}_x = \mathbf{S}_x * A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A$$

$$\mathbf{G}_y = \mathbf{S}_y * A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

Why is this an approximation of the gradient?

- Magnitude of gradient

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

- Angle

$$\Theta = \text{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

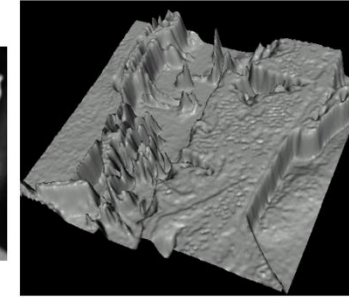
- Gradient intensities can be thresholded to detect edges

Bildgradienten

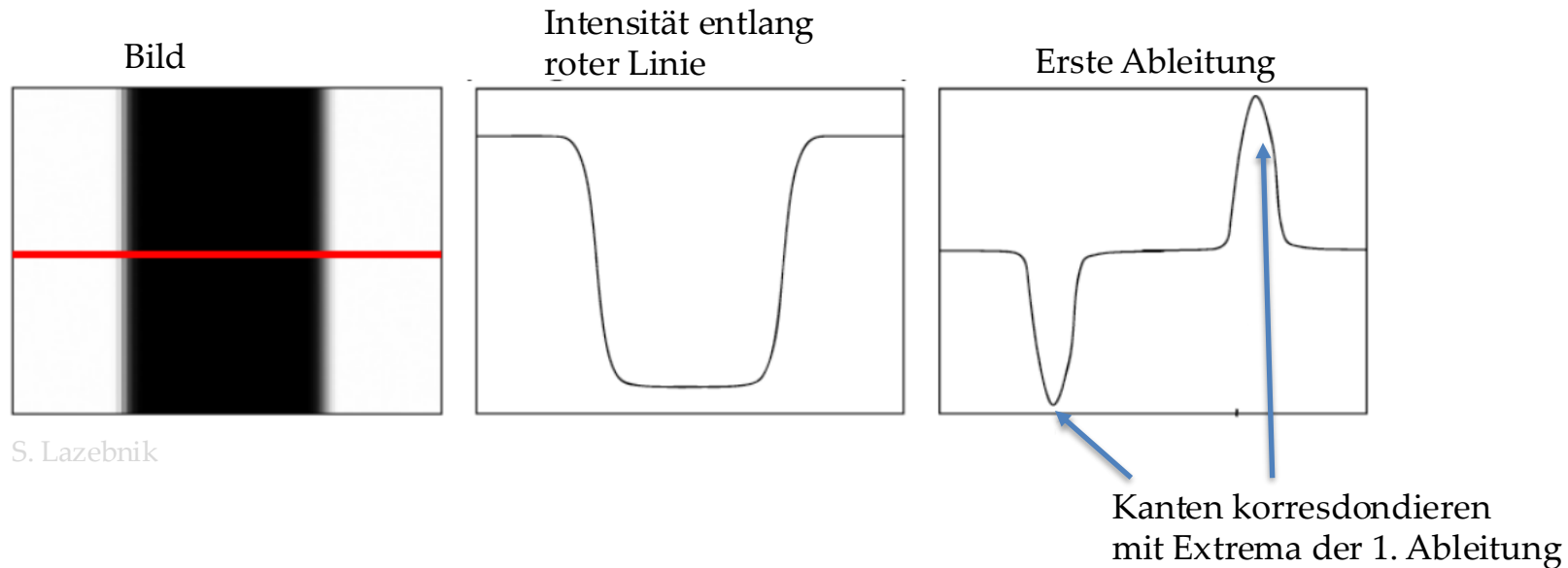
- Kanten sehen aus wie steile Abhänge



N. Snavely



- Orte plötzlicher Änderung der Bildintensität



Bildgradienten

- Berechnung der diskreten Ableitung über finite Differenzen

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

partielle Ableitung in x-Richtung

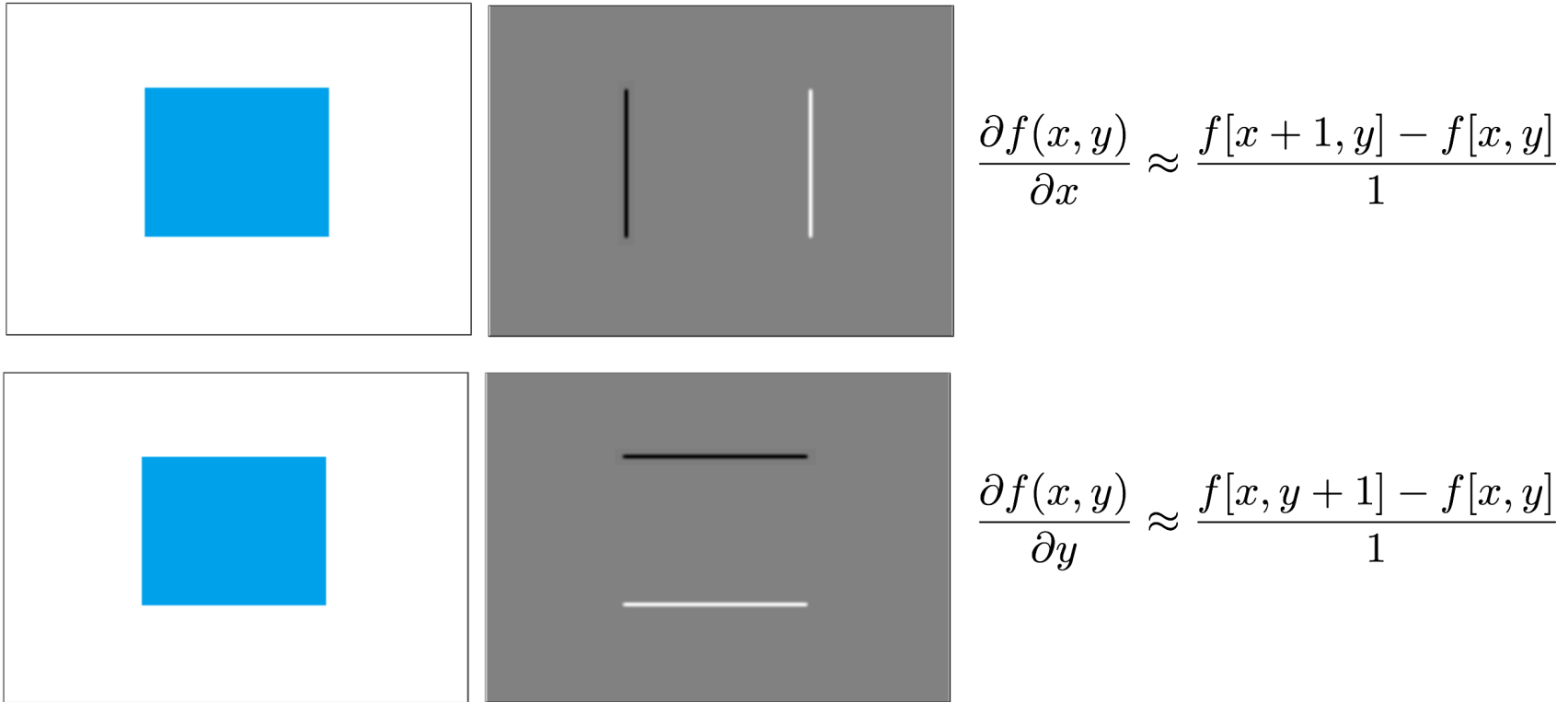
$$\frac{\partial f(x, y)}{\partial y} \approx \frac{f[x, y + 1] - f[x, y]}{1}$$

partielle Ableitung in y-Richtung

- Wie könnte dafür ein Convolution-Filter aussehen?

Bildgradienten

- Convolution-Filter $[-1,1]$ (horizontal), $[-1,1]^T$ (vertikal)



Fidler, Univ. Toronto

Finite Differenzen Filter

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

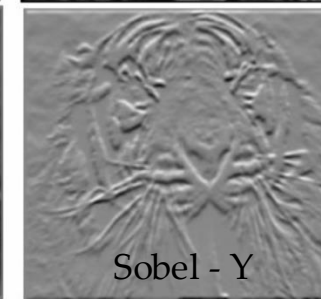
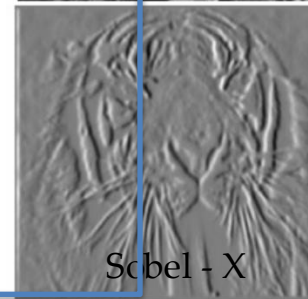
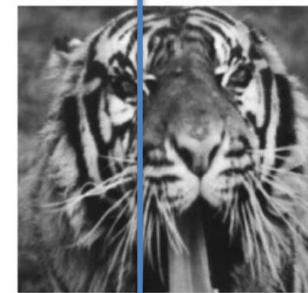
Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

K. Grauman

Demo

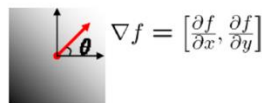
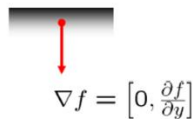
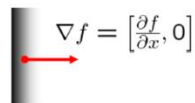
Important changes in image



S. Lazebnik

Image gradient as 2D Vektor:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$



Gradient direction:

Direction of the edge normal

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

MoG:

Length of direction vec $\|\nabla f\| = \sqrt{\frac{\partial f^2}{\partial x} + \frac{\partial f^2}{\partial y}}$

Separierbarkeit

- **Separability refers to the property that the impulse response of a two-dimensional filter (e.g. with Sobelkernel) can be represented by multiplying two one-dimensional operators**
- Second operator is applied to the intermediate result of the first operator
- **original 2D filter split into x and y kernels and applied one after the other to the original image**
- Why?
 - Saves computing time because 2D filter requires N^2 multiplications + $N^2 - 1$ additions
 - Separate variant only $2N$ multiplications and $2(N-1)$ additions
- Example Sobel:

$$\text{Sobel} \quad M_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad M_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad M_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * I = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} * (1 \quad 0 \quad -1) * I$$

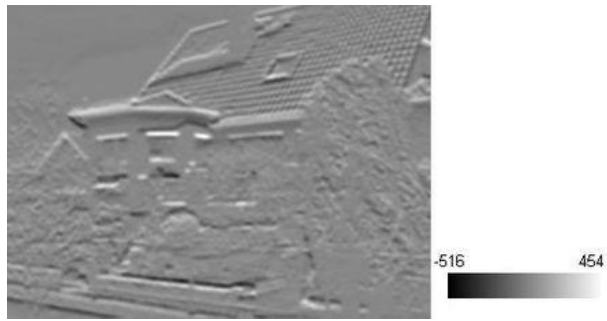
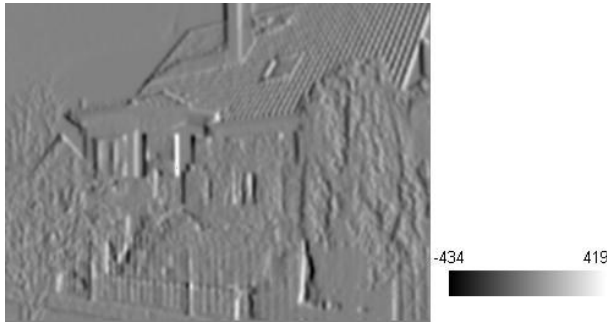
Canny-Edge Detection

Canny Edge

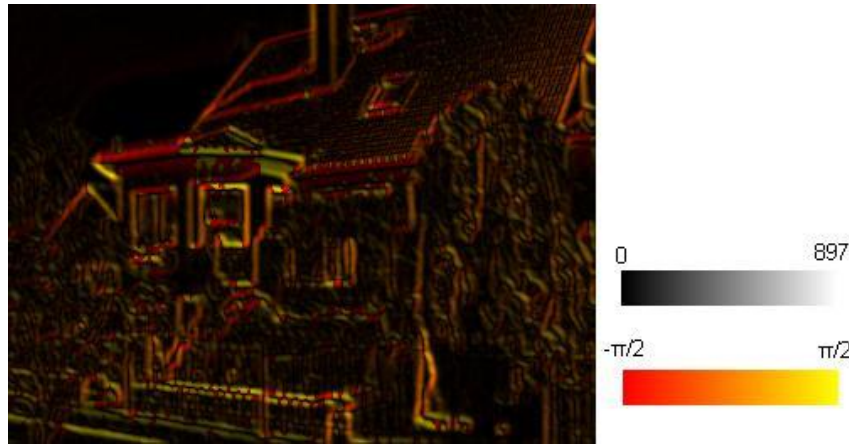
1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image



© wikipedia



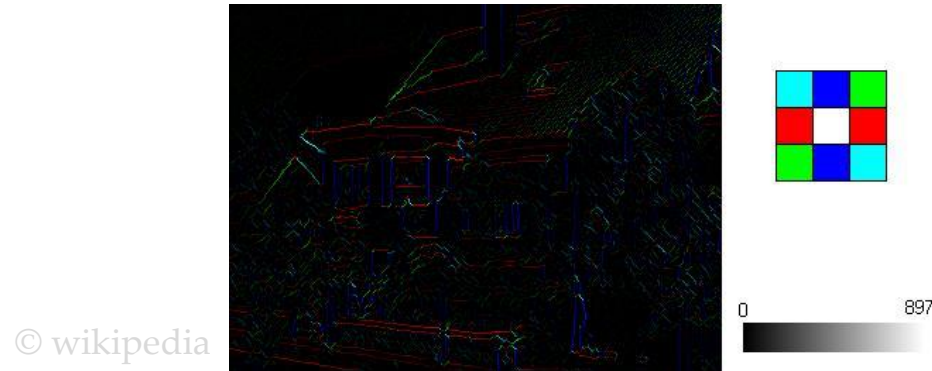
© wikipedia



© wikipedia

Canny Edge

3. Apply non-maximum suppression to get rid of spurious response to edge detection



4. Track edge by hysteresis $T_1 < T_2$: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.



© wikipedia

Questions?
Take away message