BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

# Learning from Images

## Image formation and Image Processing Basics

Master DataScience
Winter term 2019/20

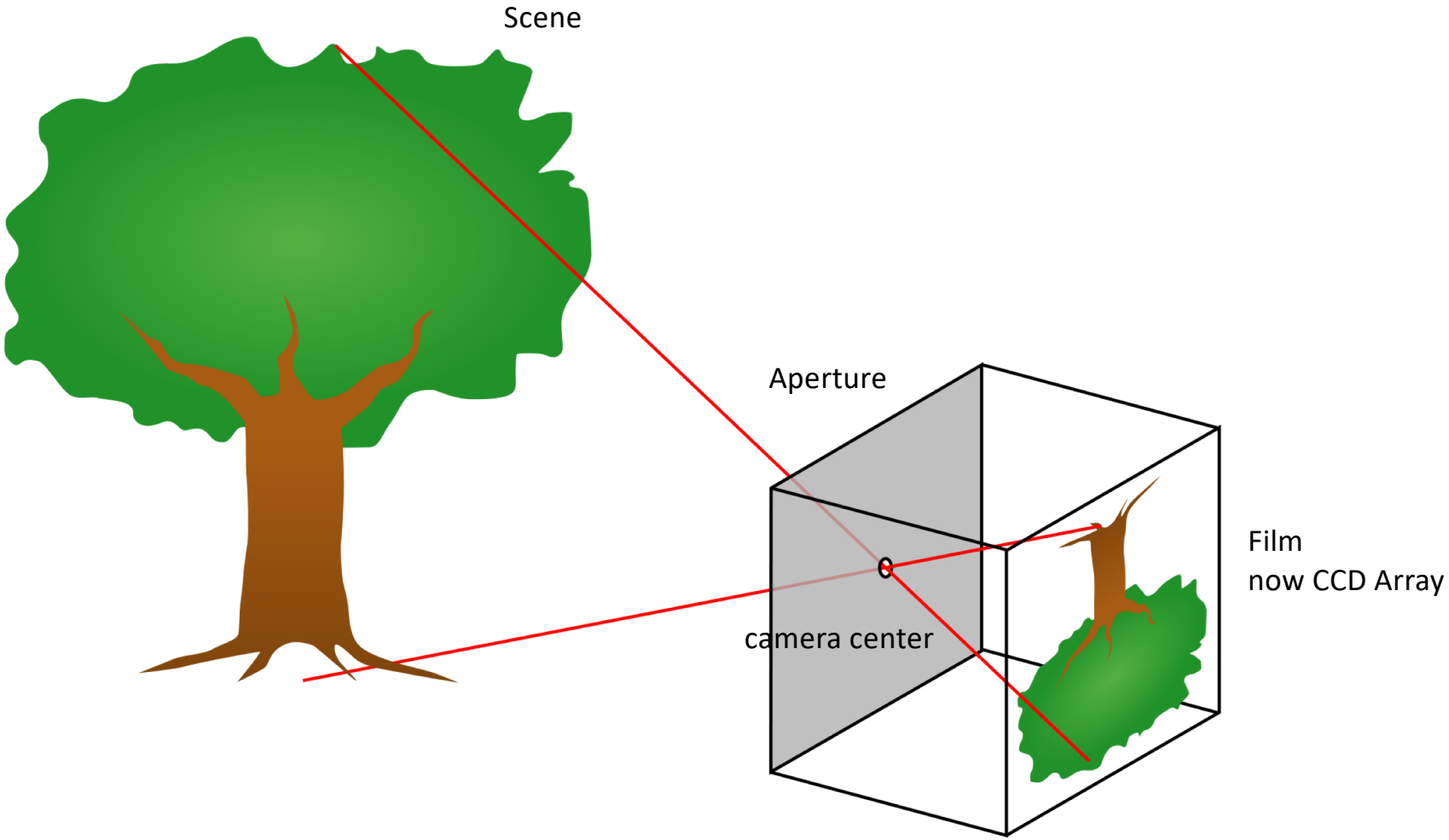
Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

# Goals for today

- Image formation
- Image representation
- Image processing
  - Filter
  - Algorithms, e.g. Canny-Edge, Color Quantization

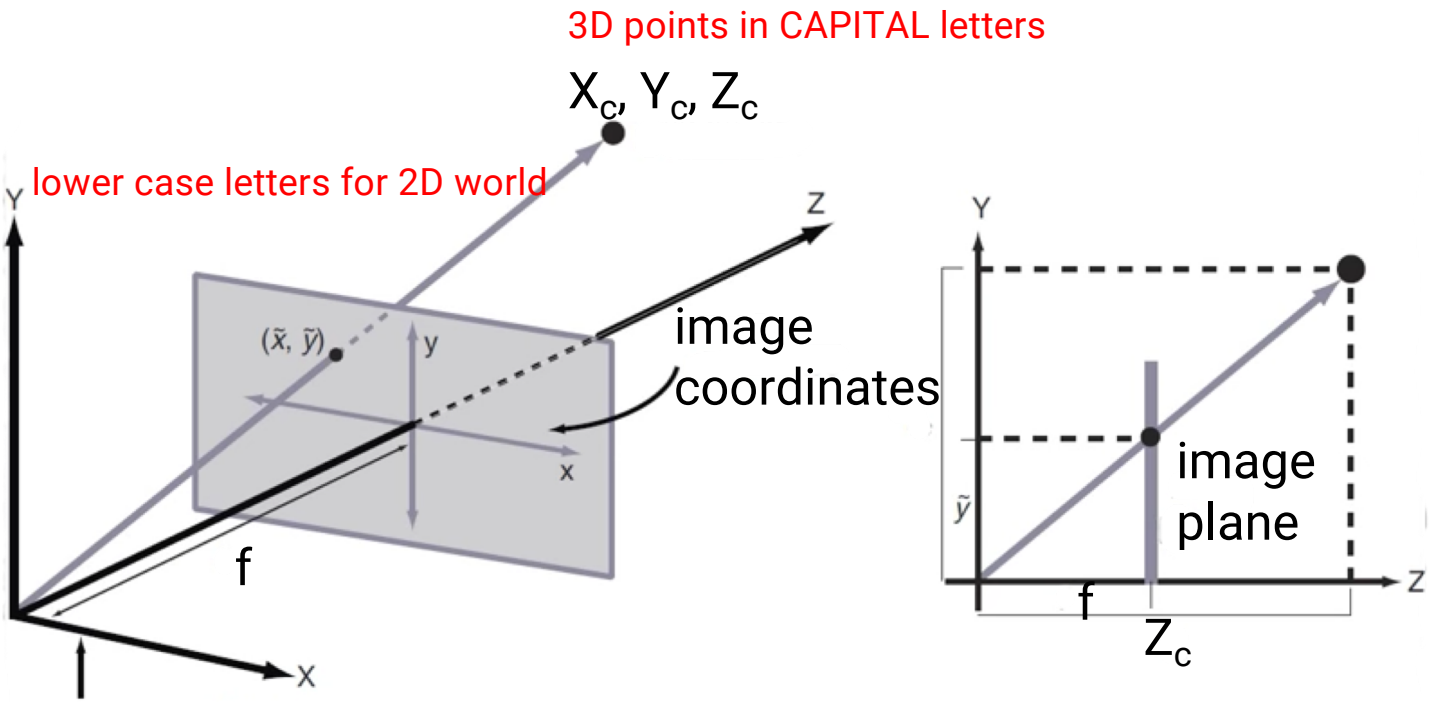
- ***Assignment 1 is out***

# Image formation




wikipedia.org

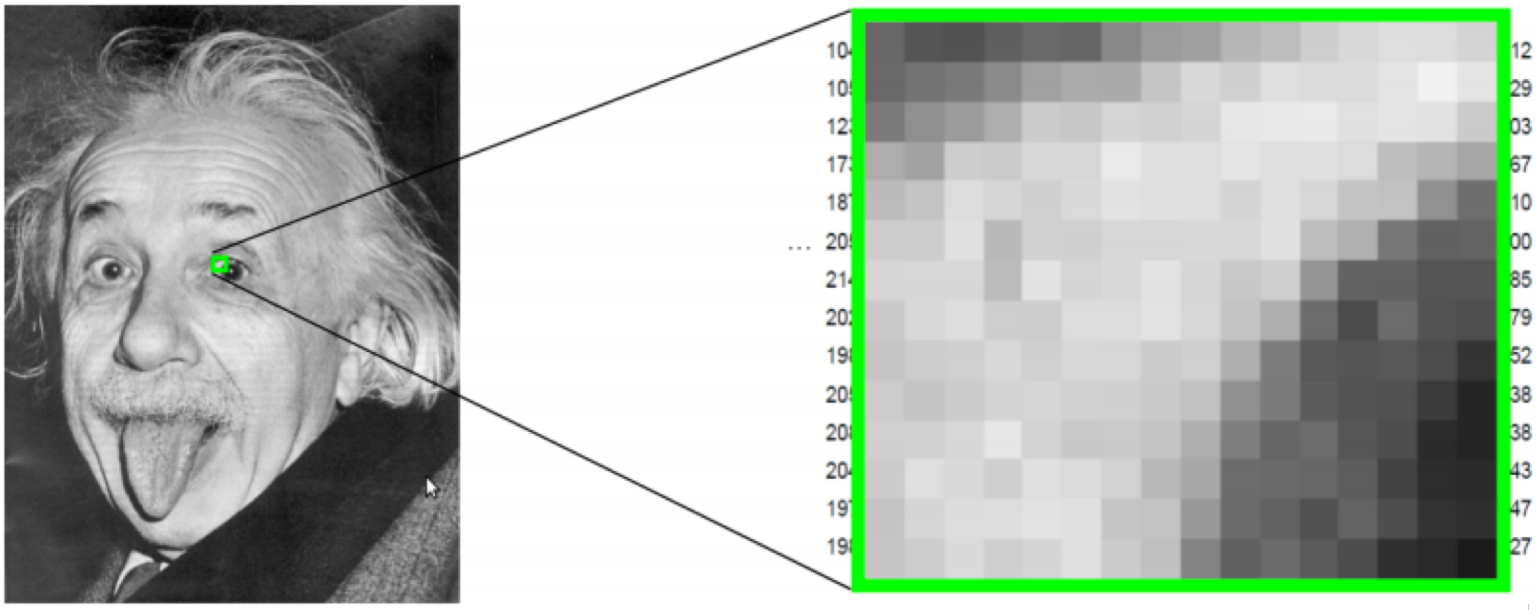
# Image formation




camera coordinate
system (aligned with image plane)

# Image representation

# Image representation




D. Schlesinger, TU Dresden

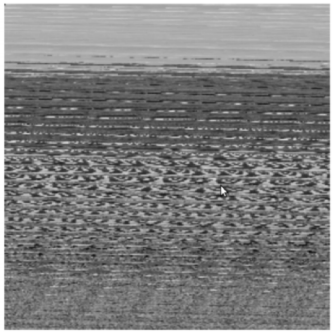
## Matrices

or vectors

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

**Pixel's intensity value**

# Images are Matrices

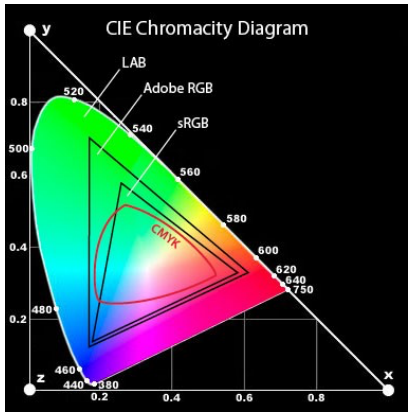


D. Schlesinger, TU Dresden

- 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

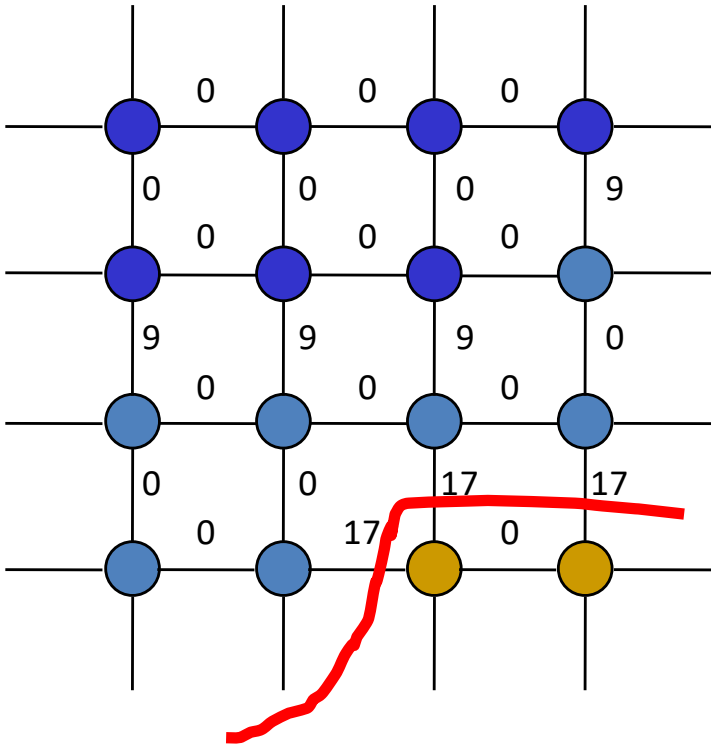


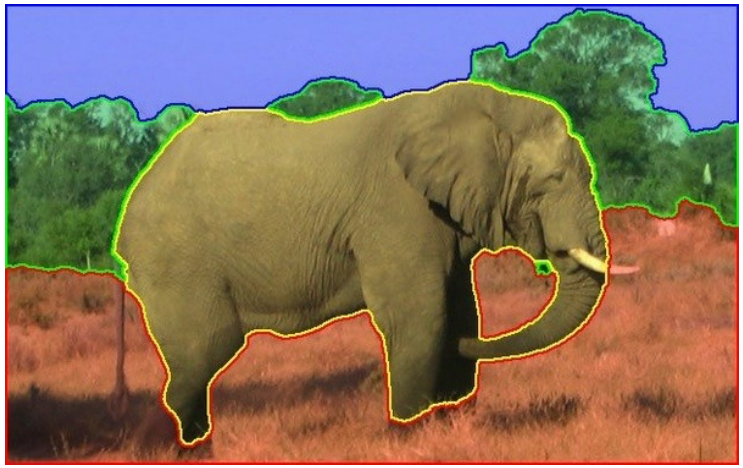

https://www.photo.net/learn/using-lab-color-adjustments/

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

# Images as graphs

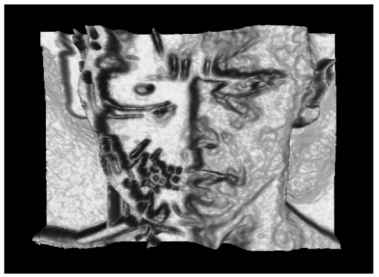
- Pixel are nodes with 4- (or 8-) neighbors
- Edges are weights
  - e.g. color distances between pixels
- Graph is a grid
  - nodes can be labeled for segmentation etc.
- Algorithmen:
  - Graph Cut
  - Conditional Random Fields (CRF)



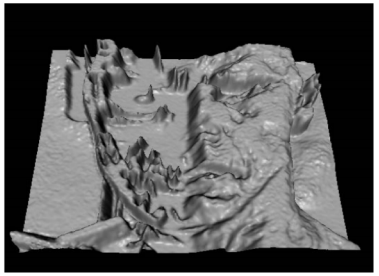



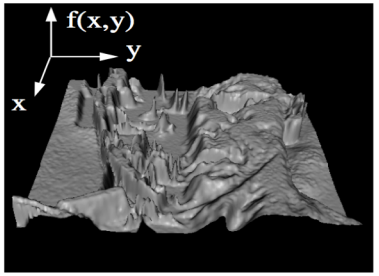
# Images are functions

- convex, continous, differentiable
- *I(x,y)* are intensities at location *(x,y)*



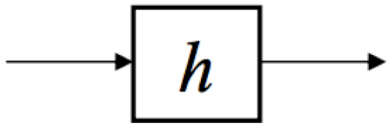
- change of signal through transformation function *h*






N. Snavely

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




Invert, Histogram, Contrast
etc.

Fredo Durand, MIT

# Fourier Transform

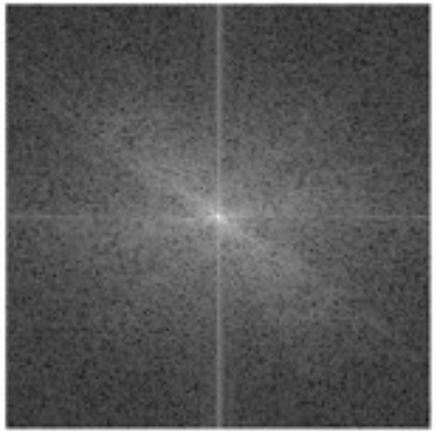
$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

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

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

Image

Fourier transform



Image

Low pass filtered

**Compression**

Original

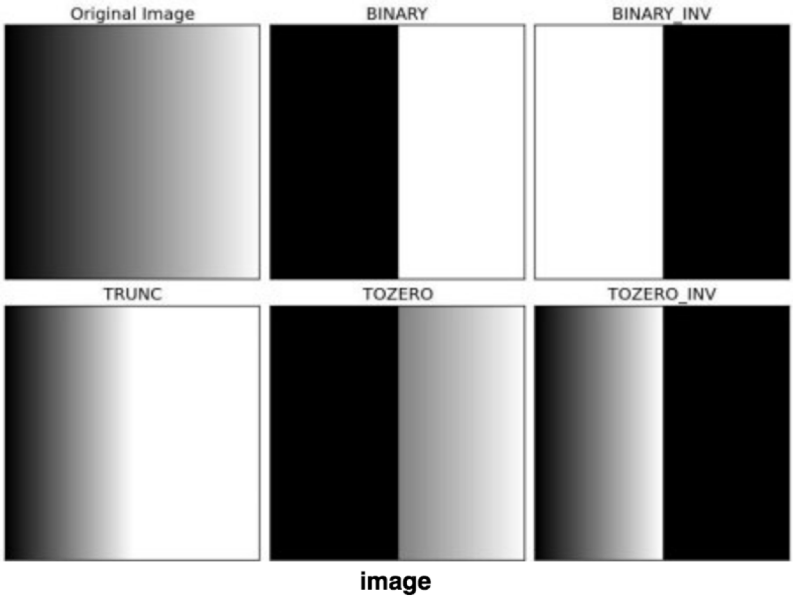
JPEG 1:64
DCT

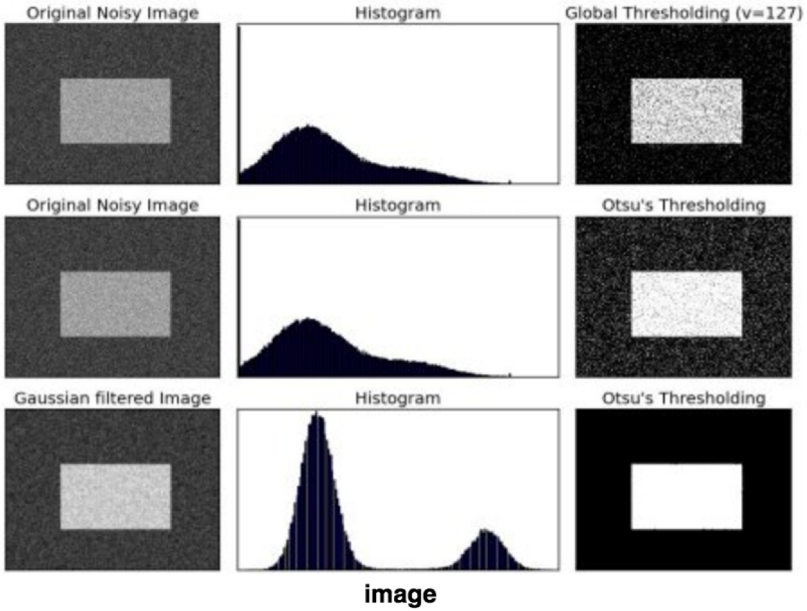
JPEG2000 1:64
DWT

# Image Processing
# What is OpenCV?

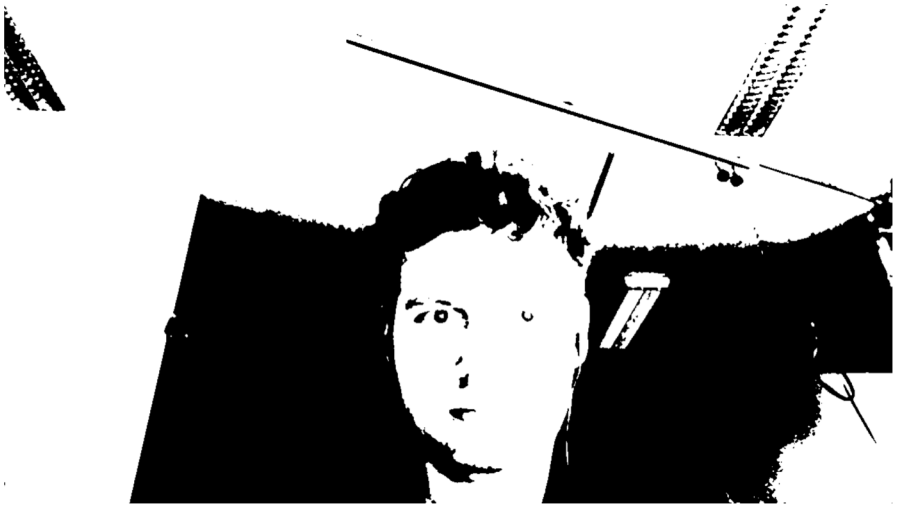
# Image processing

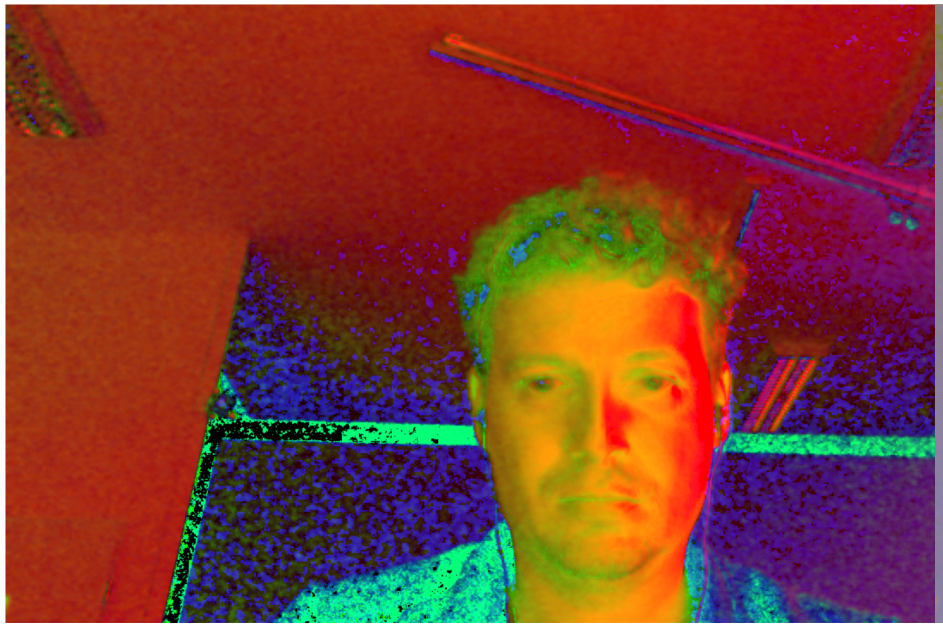
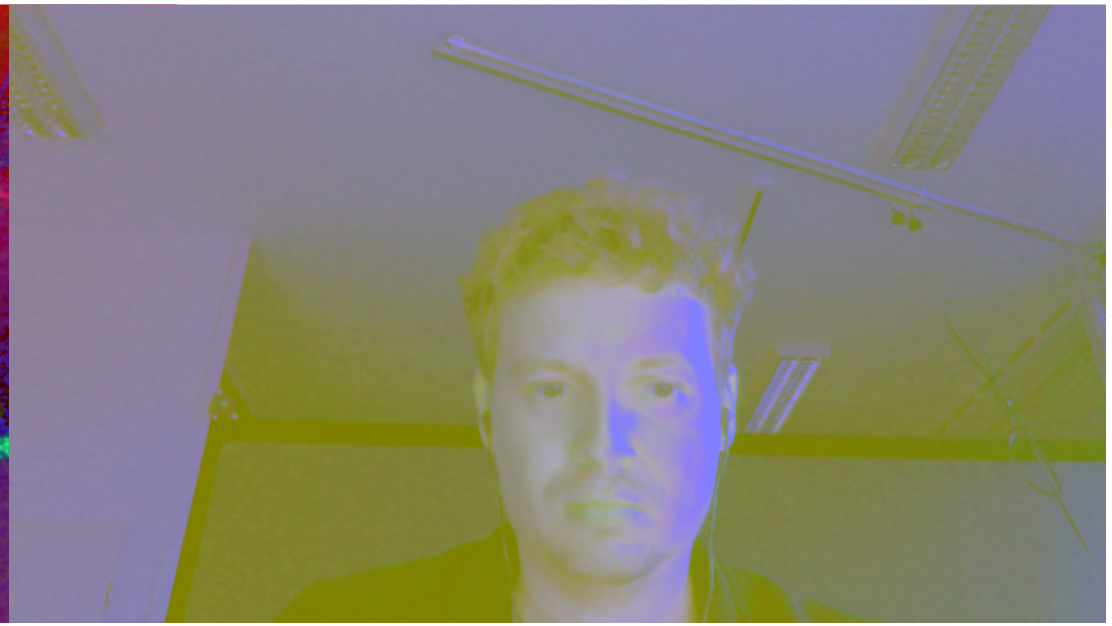
- Thresholding
- Changing Colorspaces
- Convolution + Image Gradients
- Canny Edge Detection
- Color Quantization with k-means



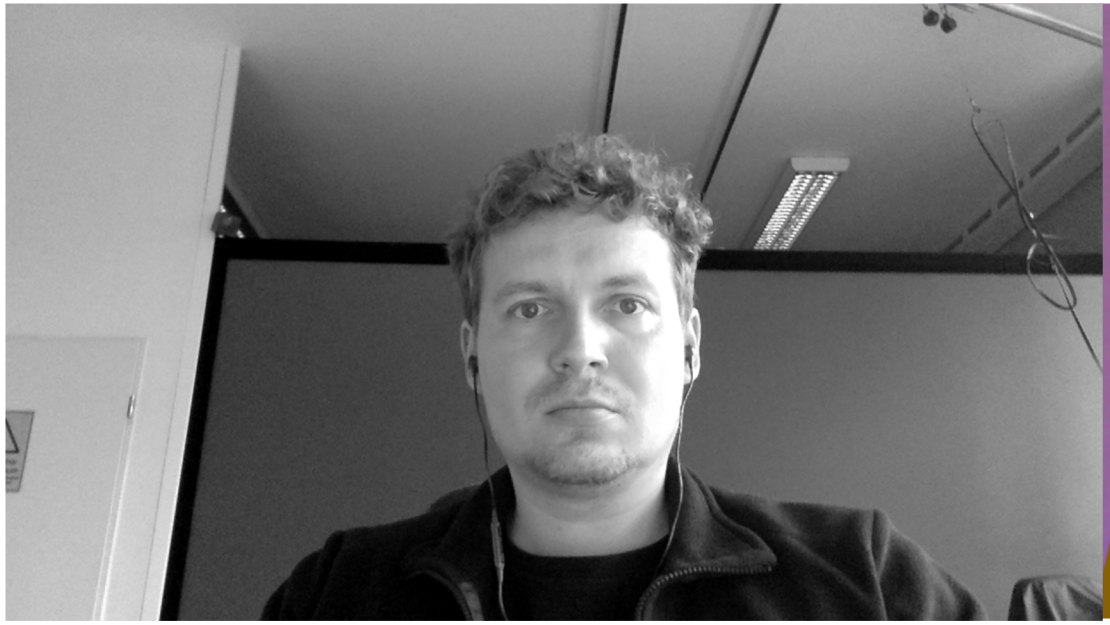
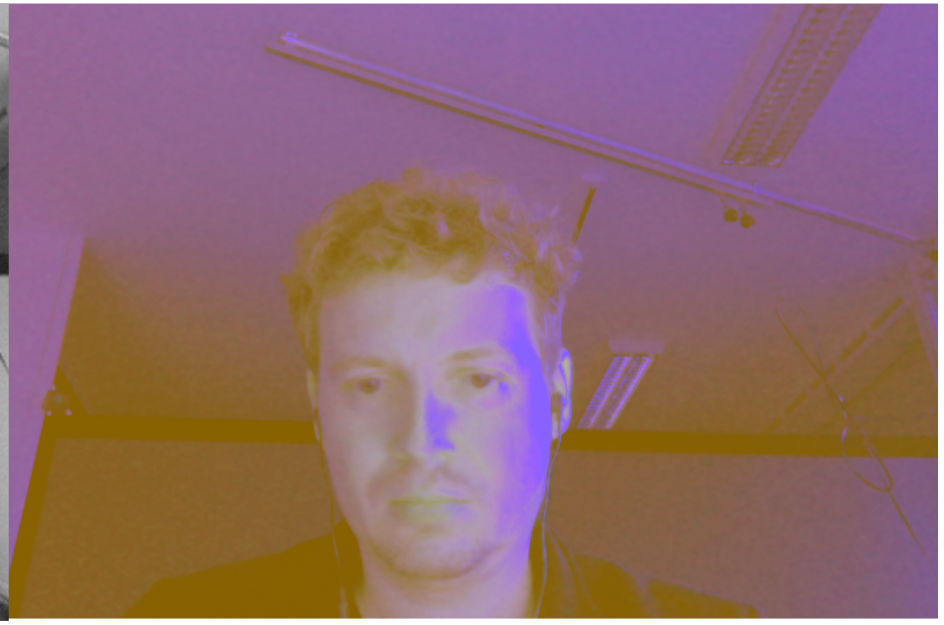




# Thresholding



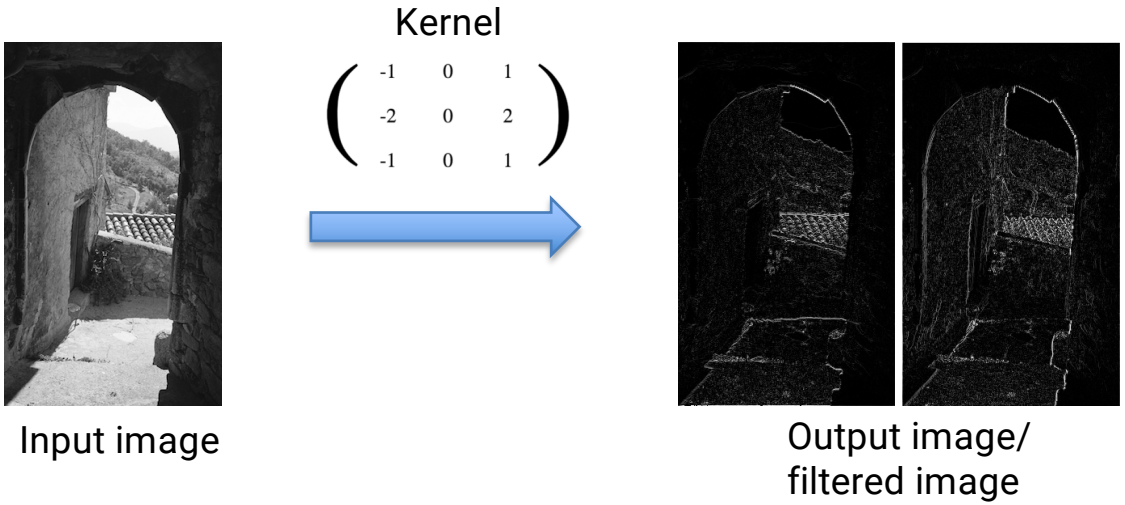
# Colorspaces

# Convolution

# Convolution

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



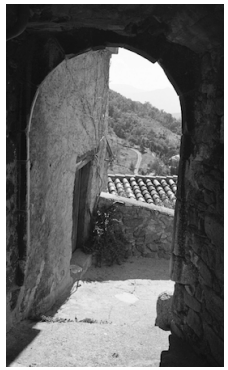

- Compute new intensity value of central pixel as weighted sum of its neighbors

# Convolution

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

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

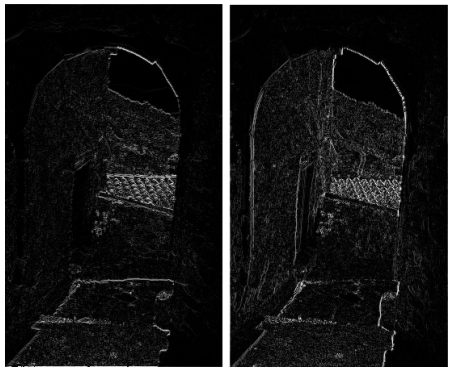
I(x,y)



Input images

g(u,v,)

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$



Output image/
filtered image

What happens at borders?

- padding
- zero

**Demo**

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

**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 a approximation of the gradient?**

- Magnitude of gradient

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

- Angle

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

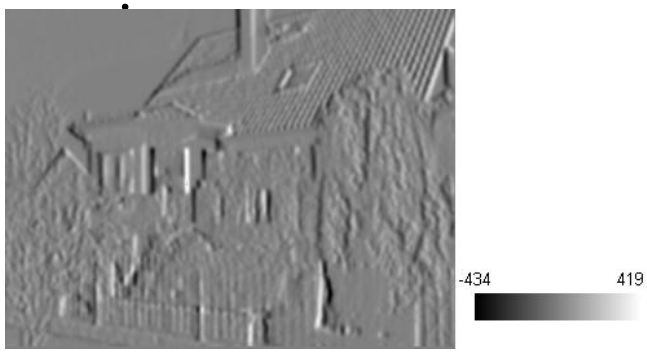
- **Gradient intensities can be thresholded to detect edges**

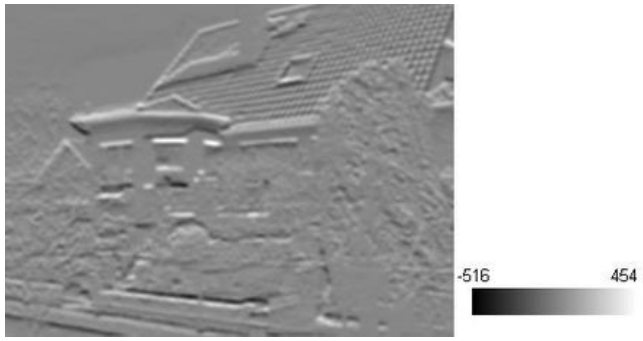
# Canny-Edge Detection

# Canny Edge

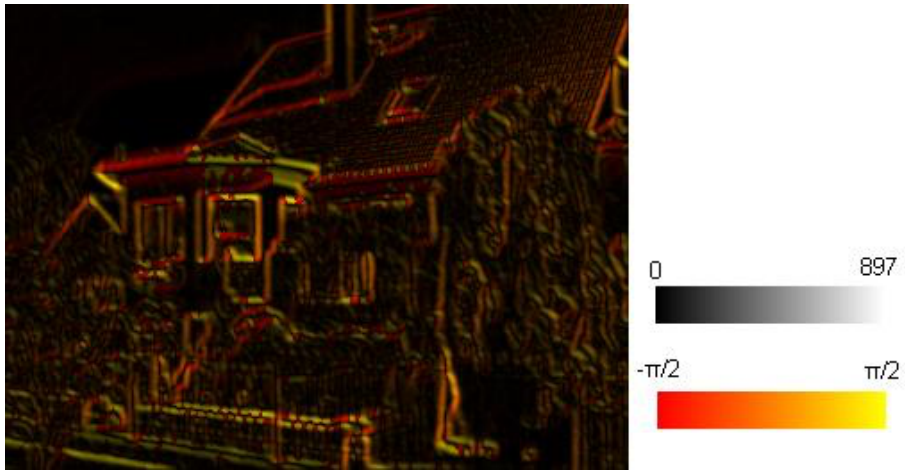
1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the

© wikipedia



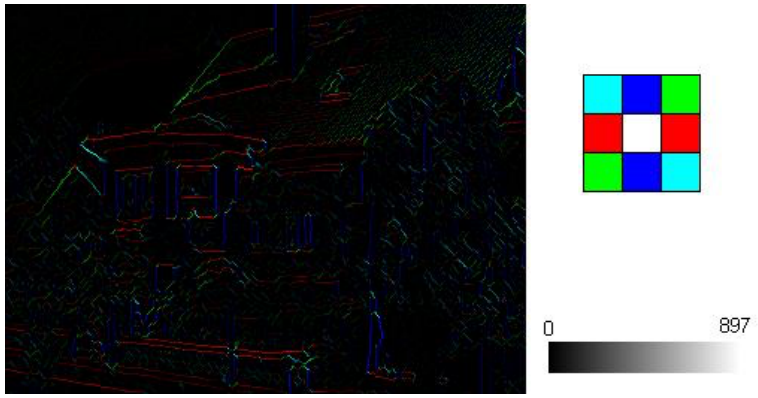

© wikipedia




© wikipedia

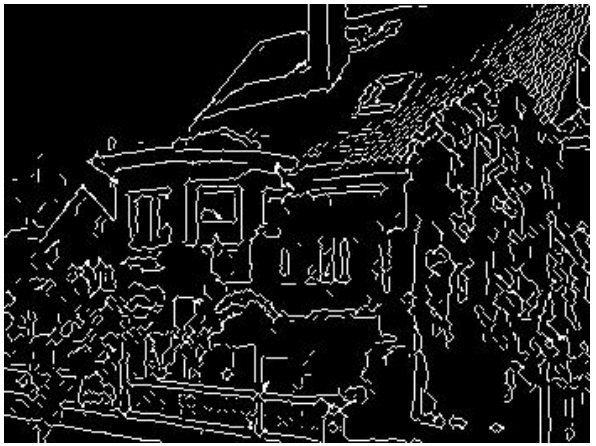
# Canny Edge

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



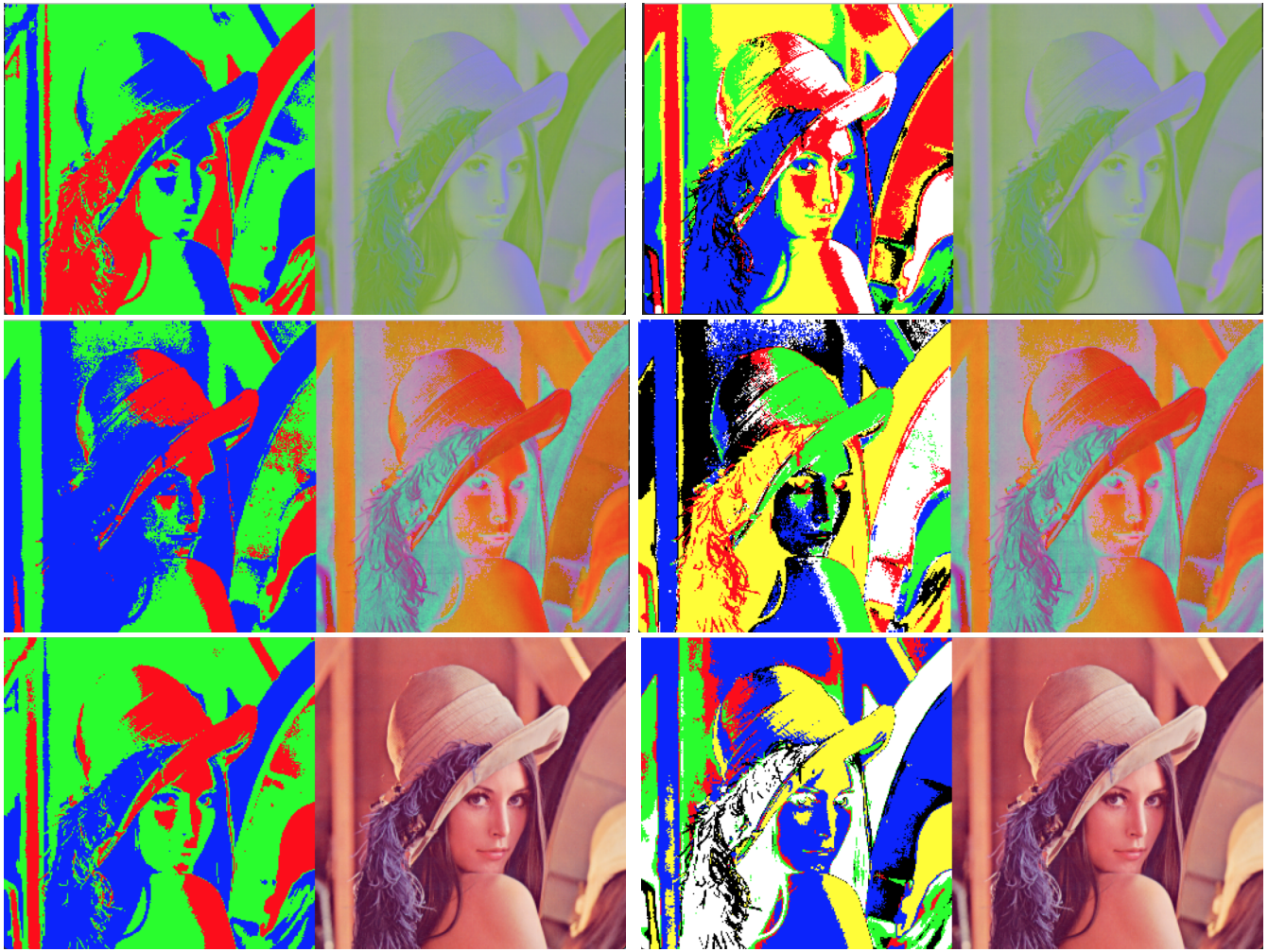



4. Track edge by hysteresis $T_1 < T_2$: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.



# Color Quantization

Original k=4 k=16 k=32 k=64



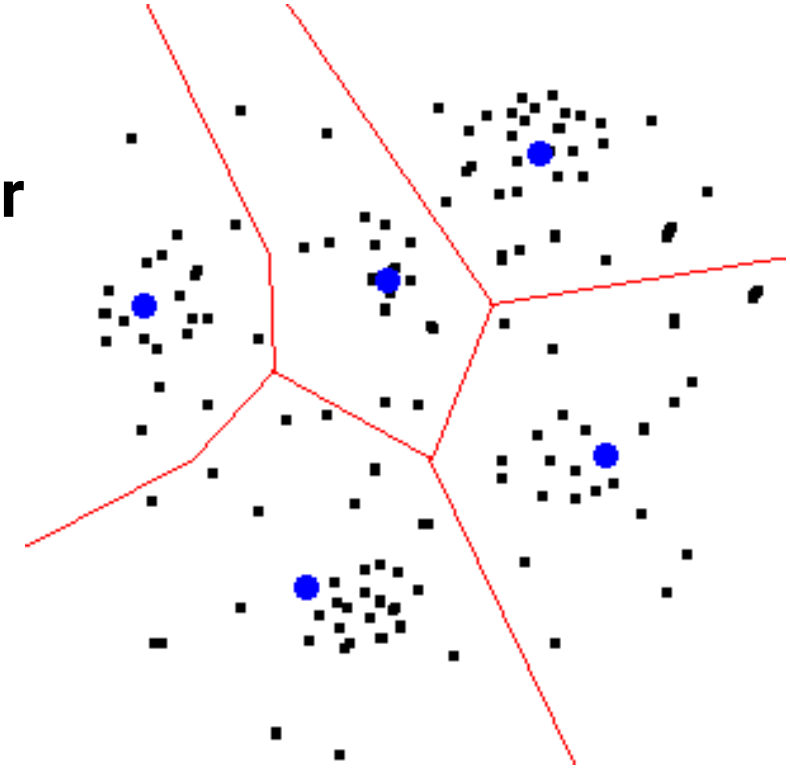
(a) (b)

# k-means

- important **clustering method** in CS
- partitions ***N* data points** in ***k* clusters *S* = {$S_1$, $S_2$, … $S_k$}**
- 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 )**
- also known as **Lloyd's Algorithm**

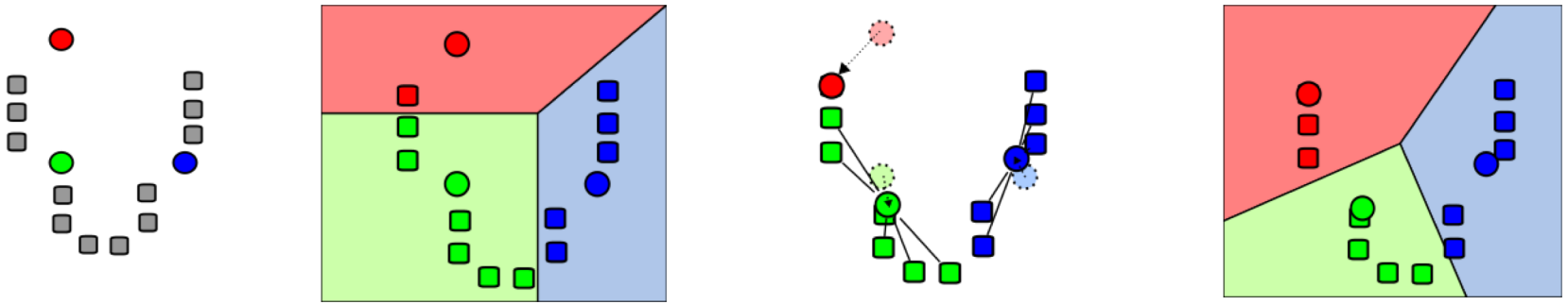


http://mnemstudio.org/

# 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)



# Questions?
# Take away message