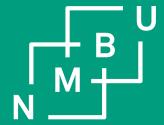
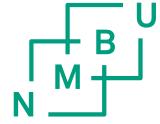


INF250

Digital Images



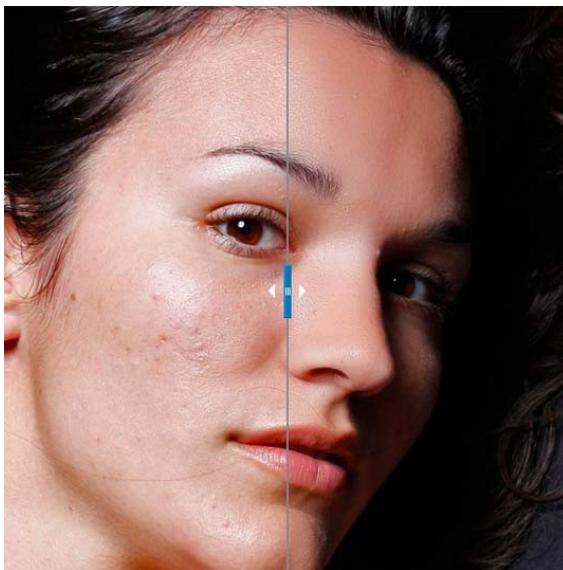


Introduction

- Chapter 1,2 in book
- Fundamental principles of digital imaging
- Demonstrations

About INF250

- Image Processing has many aspects
 - **Computer Scientists/Engineers** develop tools (e.g. photoshop)
 - **Requires** knowledge of maths, algorithms, programming
 - **Artists** use image processing tools to modify pictures
 - **DOES NOT** require knowledge of maths, algorithms, programming



Example: Portraiture photoshop plugin



Example: Knoll Light Factory photoshop plugin



Example: ToonIt photoshop plugin

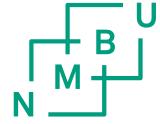
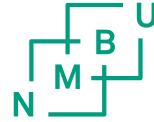


Image processing and analysis

- Image processing
 - Improvement
 - Making ready for analysis
- Image analysis
 - Properties
 - Modelling of nature
 - Classification of images
 - Objects in images



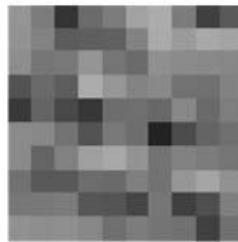
What is an Image?

- 2-dimensional matrix of Intensity (gray or color) values

Set of Intensity values

Image coordinates
are integers

$$I(u, v) \in \mathbb{P} \quad \text{and} \quad u, v \in \mathbb{N}.$$



$$F(x, y)$$

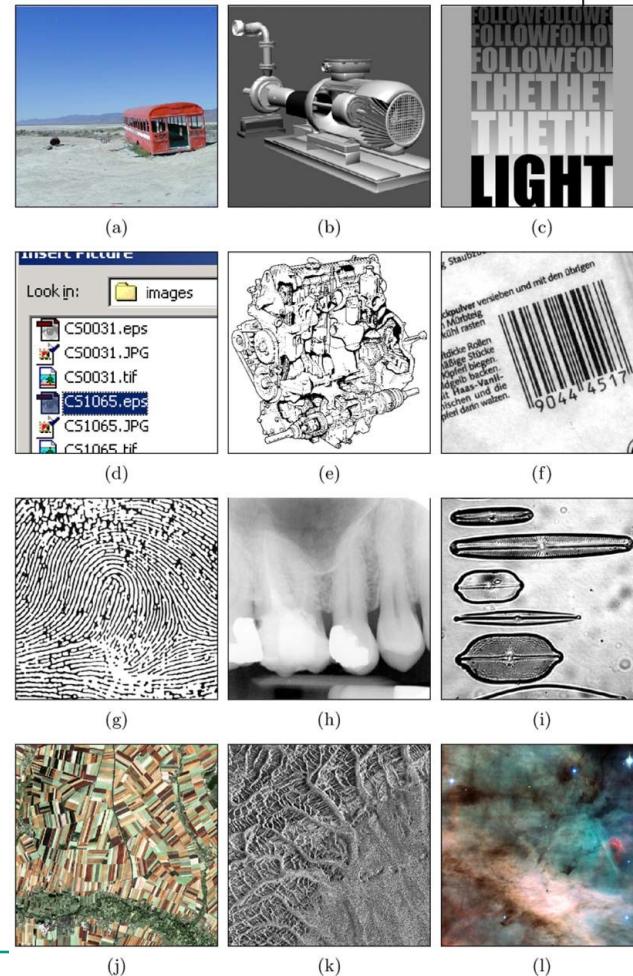


| | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 148 | 123 | 52 | 107 | 123 | 162 | 172 | 123 | 64 | 89 | ... |
| 147 | 130 | 92 | 95 | 98 | 130 | 171 | 155 | 169 | 163 | ... |
| 141 | 118 | 121 | 148 | 117 | 107 | 144 | 137 | 136 | 134 | ... |
| 82 | 106 | 93 | 172 | 149 | 131 | 138 | 114 | 113 | 129 | ... |
| 57 | 101 | 72 | 54 | 109 | 111 | 104 | 135 | 106 | 125 | ... |
| 138 | 135 | 114 | 82 | 121 | 110 | 34 | 76 | 101 | 111 | ... |
| 138 | 102 | 128 | 159 | 168 | 147 | 116 | 129 | 124 | 117 | ... |
| 113 | 89 | 89 | 109 | 106 | 126 | 114 | 150 | 164 | 145 | ... |
| 120 | 121 | 123 | 87 | 85 | 70 | 119 | 64 | 79 | 127 | ... |
| 145 | 141 | 143 | 134 | 111 | 124 | 117 | 113 | 64 | 112 | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

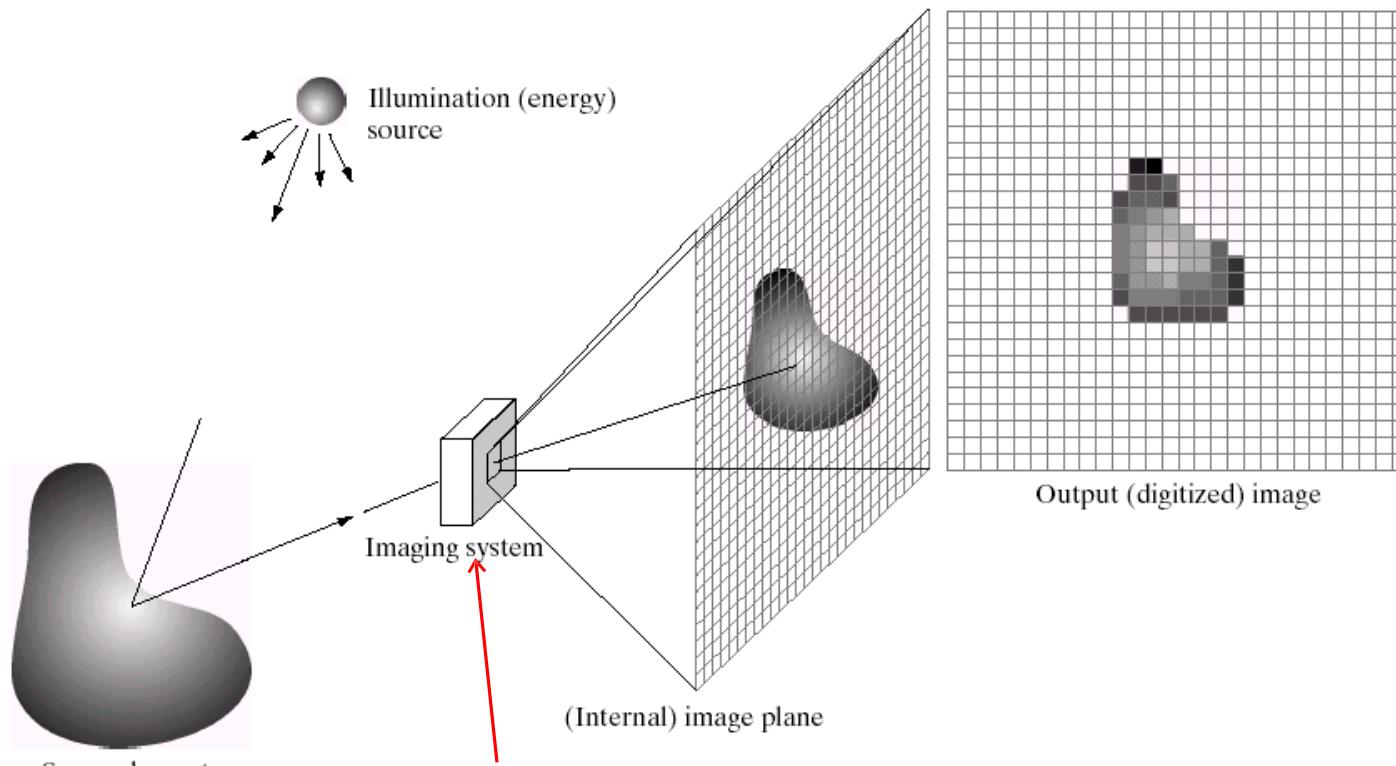
$$I(u, v)$$

Example of Digital Images

- a) Natural landscape
- b) Synthetically generated scene
- c) Poster graphic
- d) Computer screenshot
- e) Black and white illustration
- f) Barcode
- g) Fingerprint
- h) X-ray
- i) Microscope slide
- j) Satellite Image
- k) Radar image
- l) Astronomical object



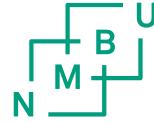
Imaging System



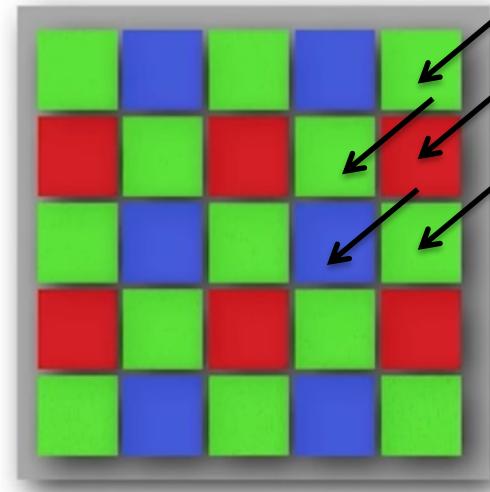
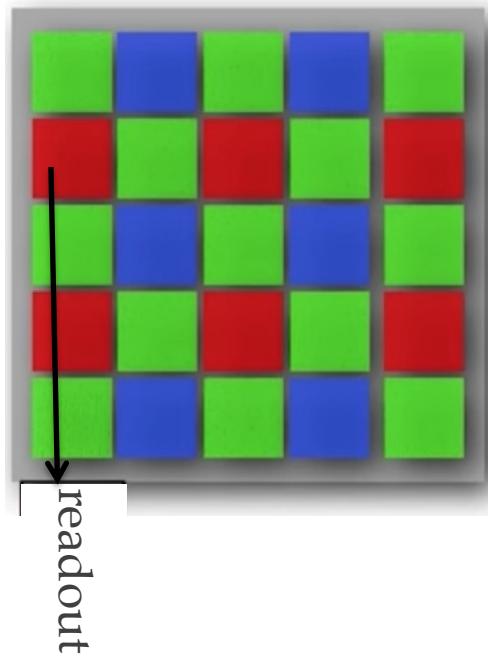
Example: a camera
Converts light to image

Ref: Gonzales and Woods

CCD and CMOS



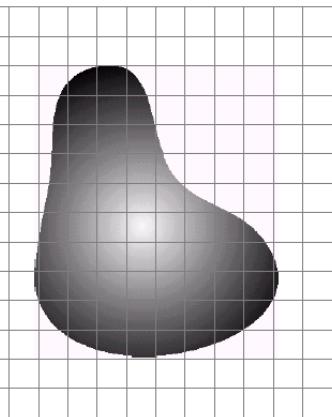
- Detector in camera is usually a Charge Coupled Device (CCD) or Complementary Metal Oxide Semiconductor (CMOS)



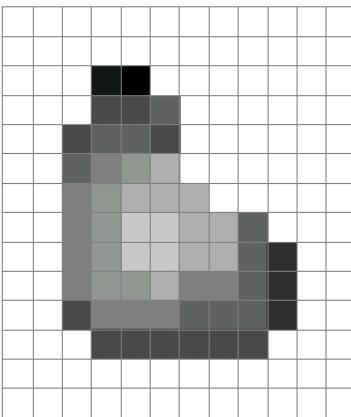
readout

- http://www.specinst.com/What_Is_A_CCD.html
- <https://www.youtube.com/watch?v=9vgtJJ2wwMA>

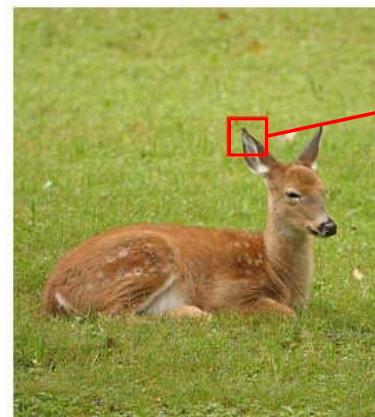
Digital Image?



Real image



Digital Image
(an approximation)



Real image



Digital Image
(an approximation)

1 pixel

Digital Image

□ Common image formats include:

- 1 values per point/pixel (B&W or Grayscale)
- 3 values per point/pixel (Red, Green, and Blue)
- 4 values per point/pixel (Red, Green, Blue, + “Alpha” or Opacity)



Grayscale



RGB



RGBA

What is image Processing?

- Algorithms that alter an input image to create new image
- Input is image, output is image



Original Image

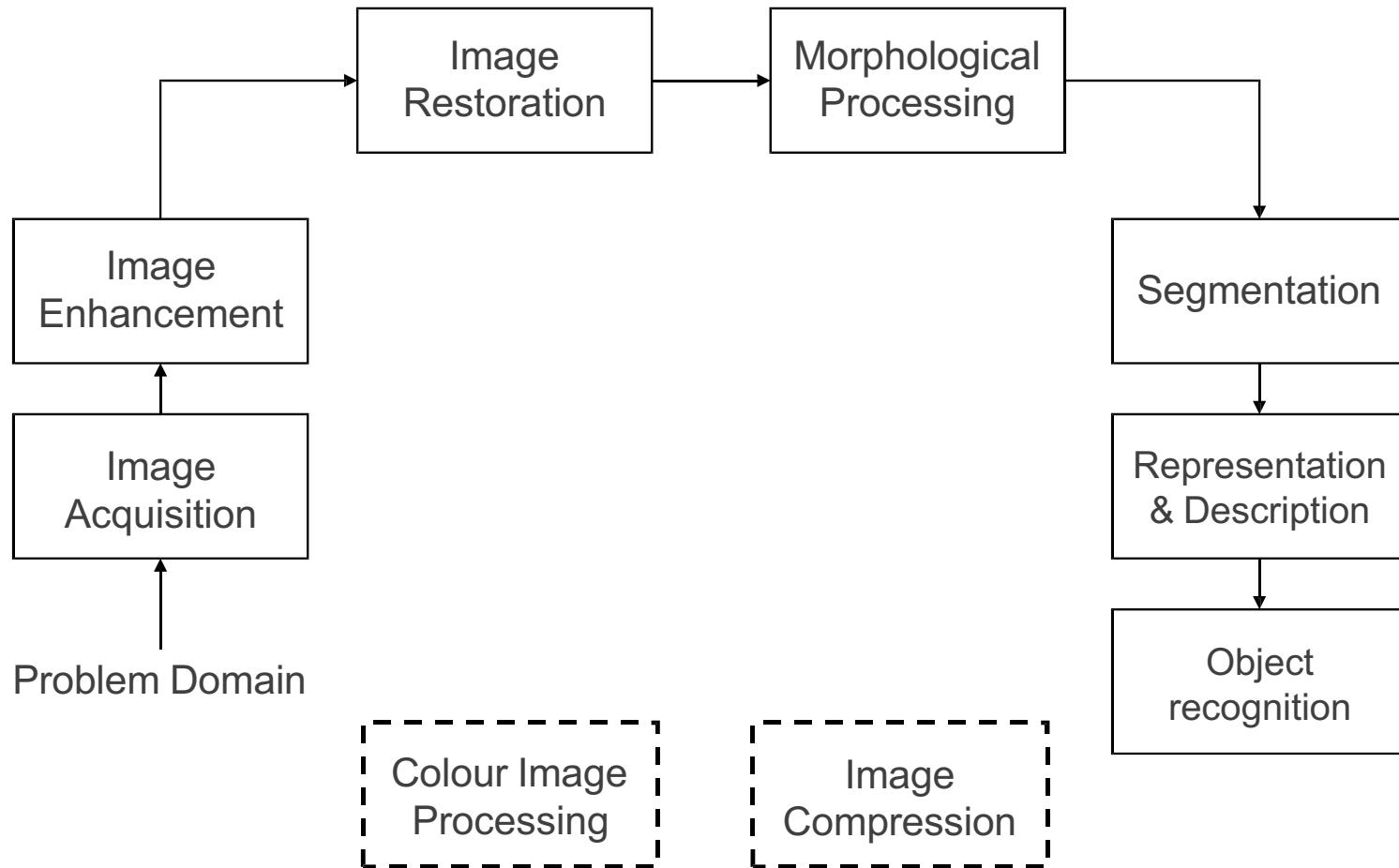
Image Processing
Algorithm
(e.g. Sobel Filter)



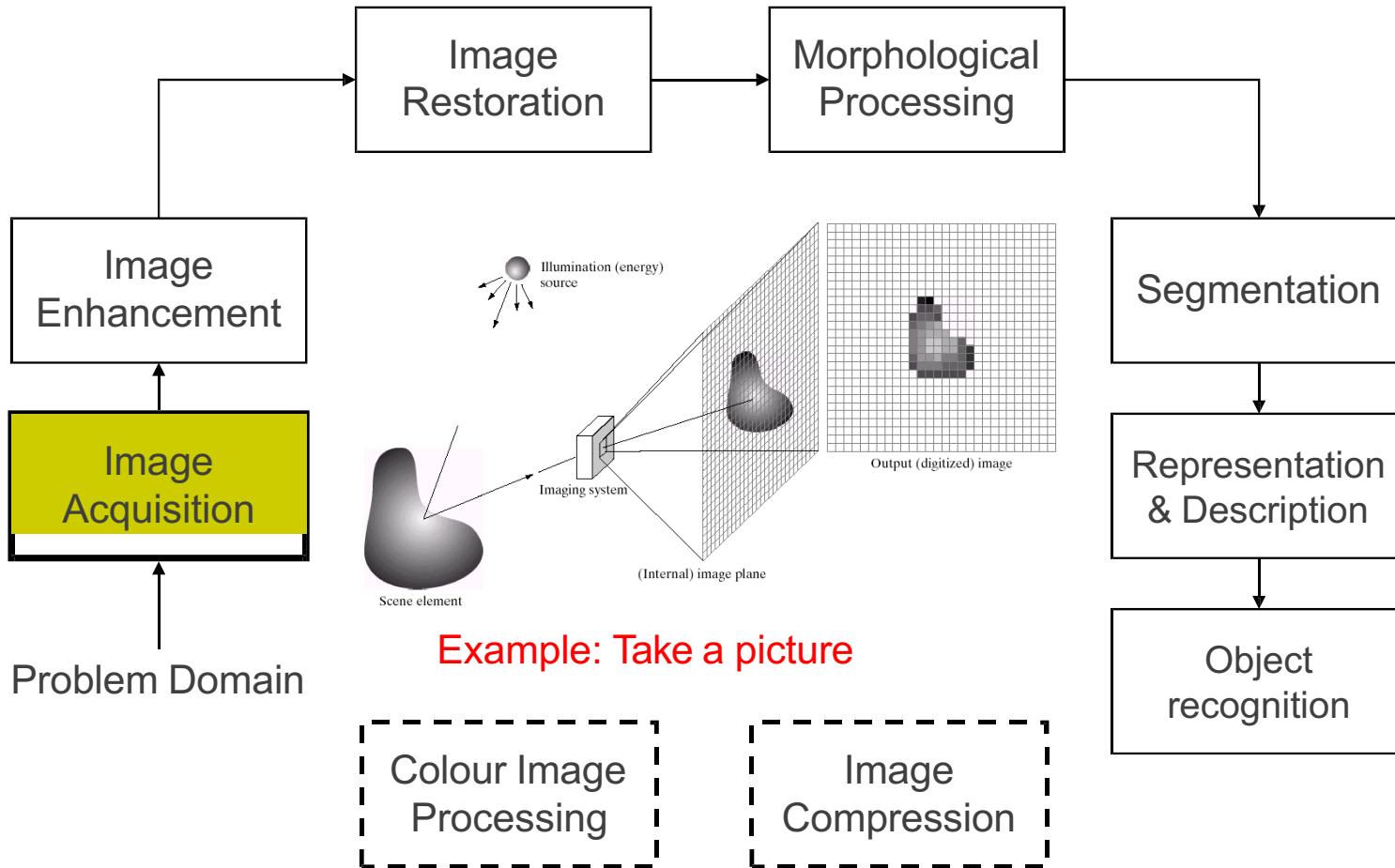
Processed Image

- **Improves an image for human interpretation in ways including:**
 - Image display and printing
 - Image editing
 - Image enhancement
 - Image compression

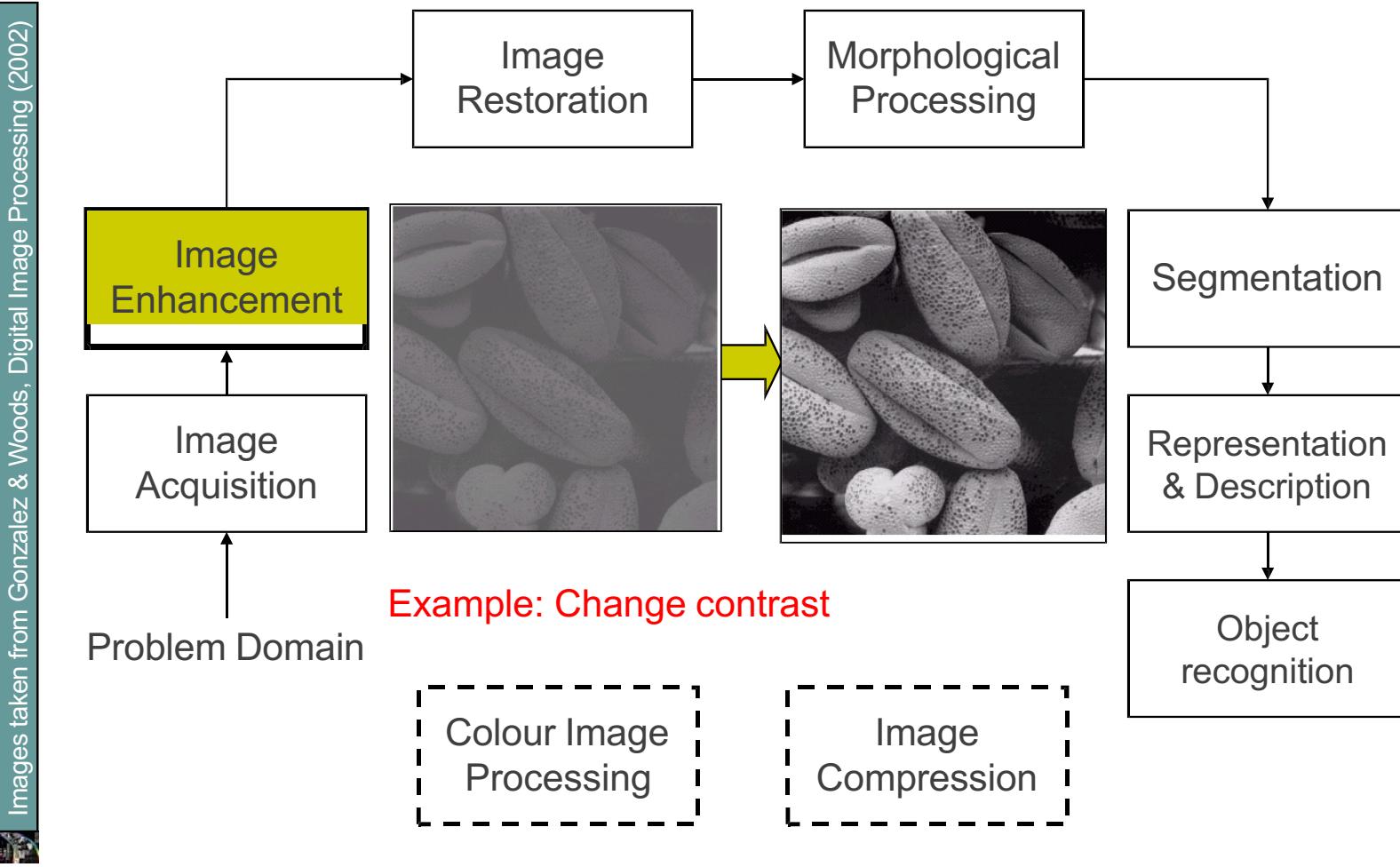
Key Stages in Digital Image Processing



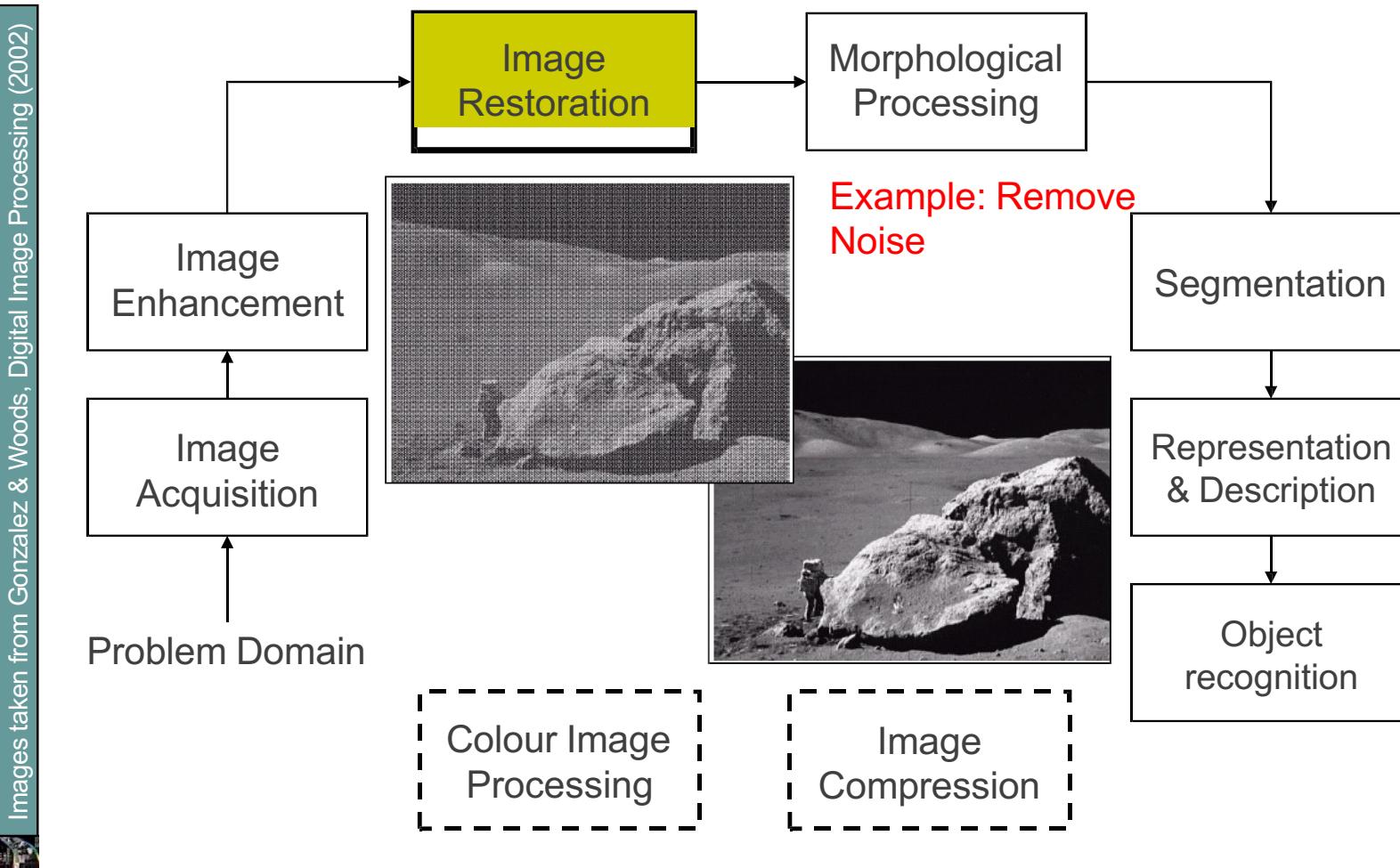
Key Stages in Digital Image Processing: Image Acquisition



Key Stages in Digital Image Processing: Image Enhancement

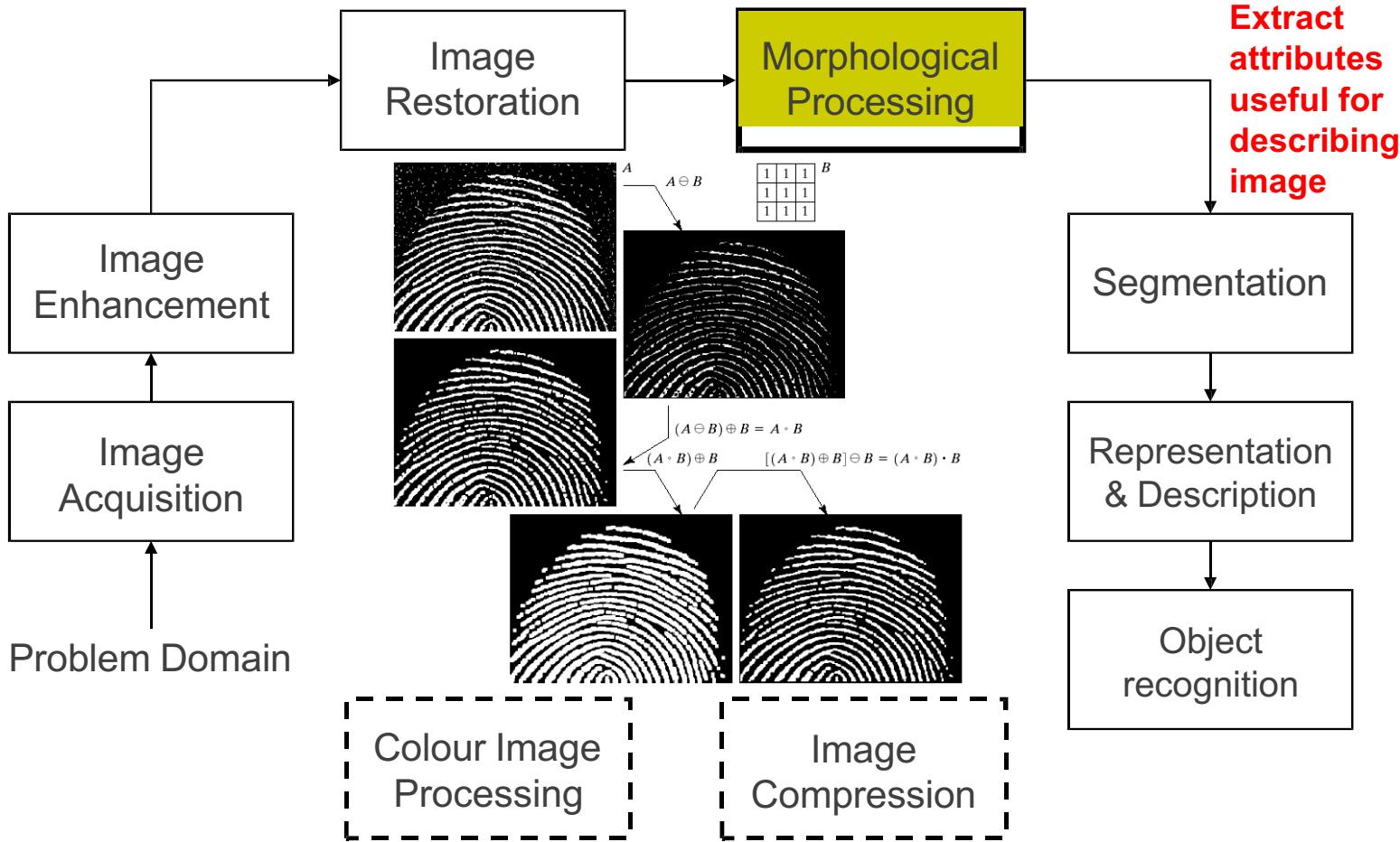


Key Stages in Digital Image Processing: Image Restoration

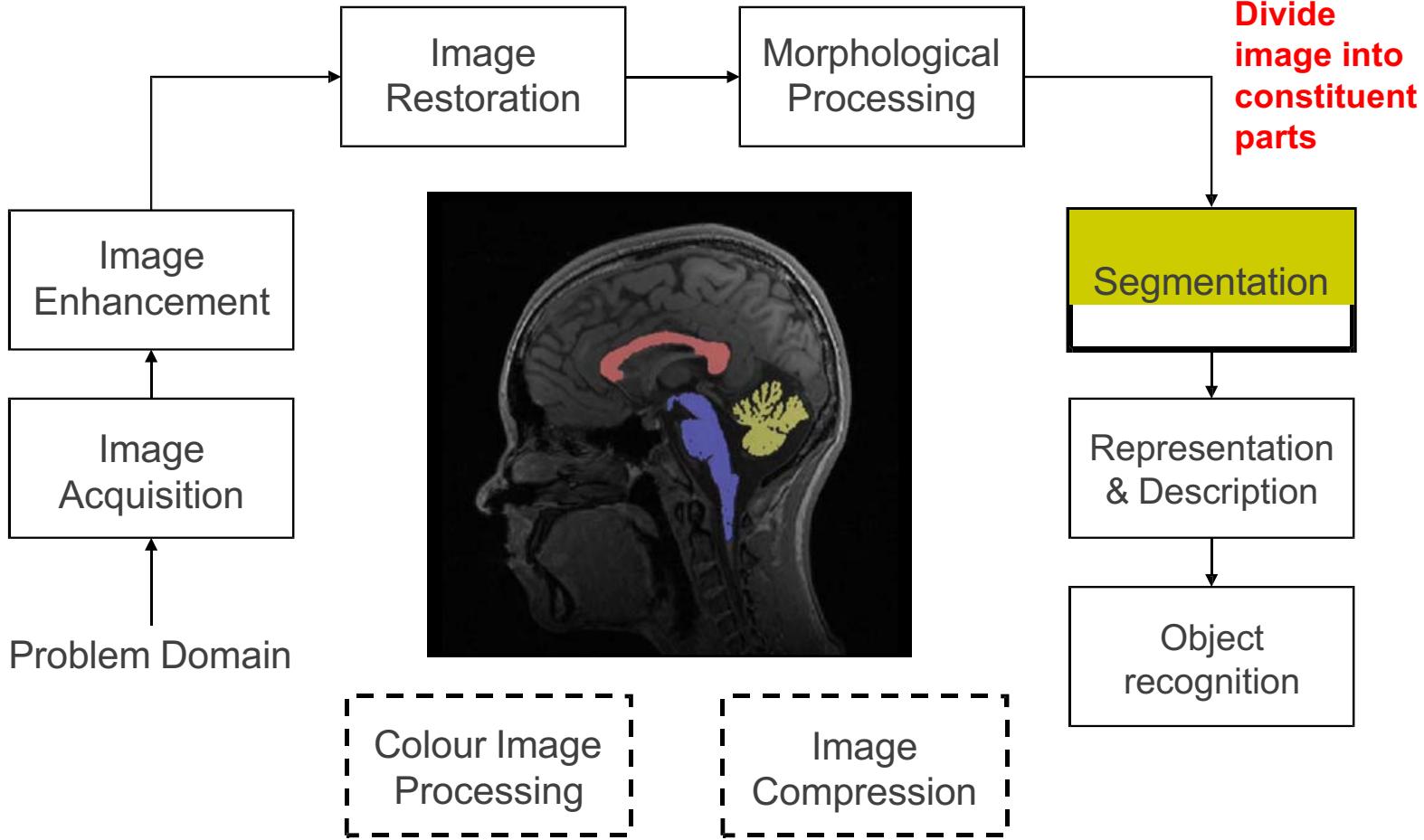


Key Stages in Digital Image Processing: Morphological Processing

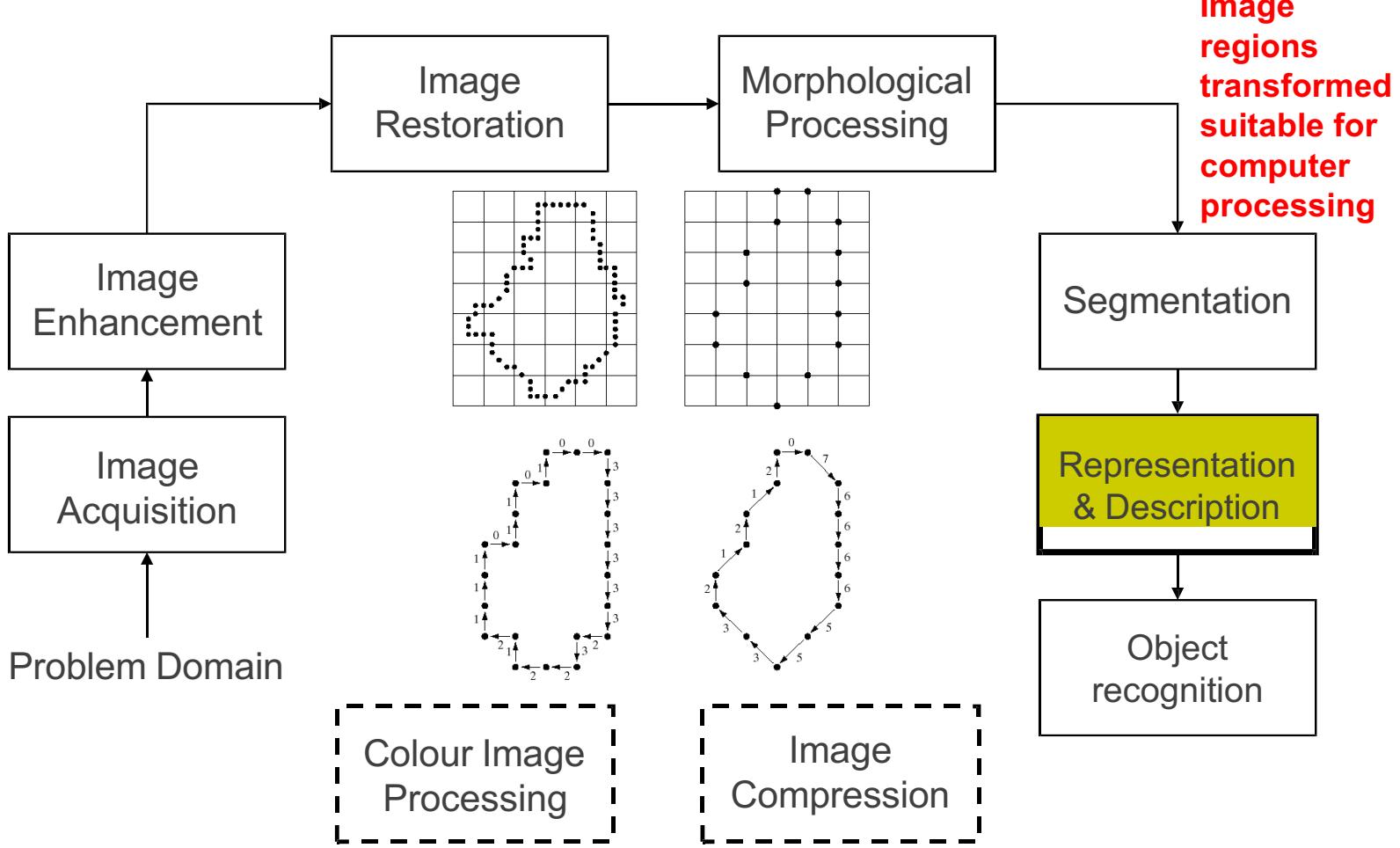
Images taken from Gonzalez & Woods, Digital Image Processing (2002)



Key Stages in Digital Image Processing: Segmentation

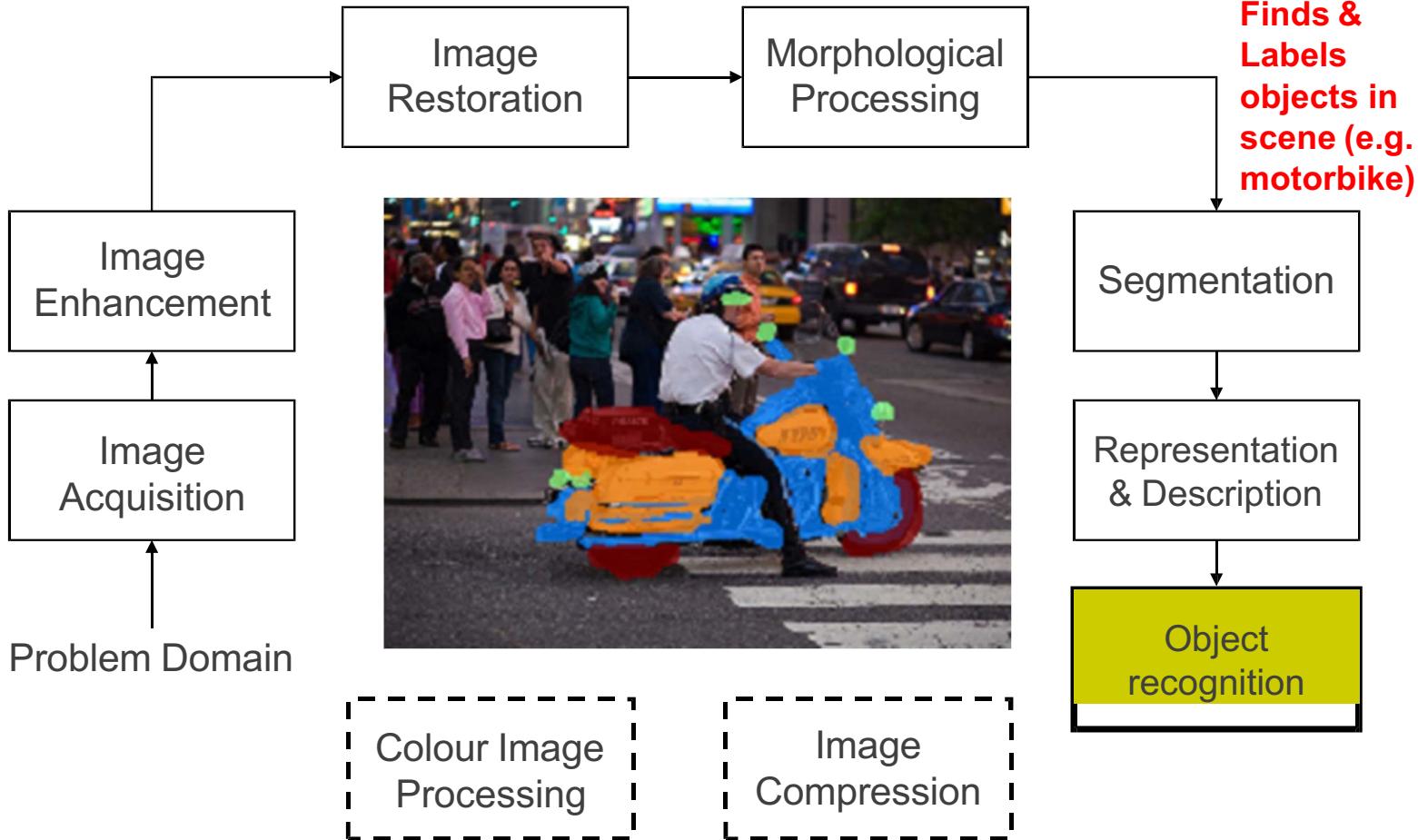


Key Stages in Digital Image Processing: Object Recognition

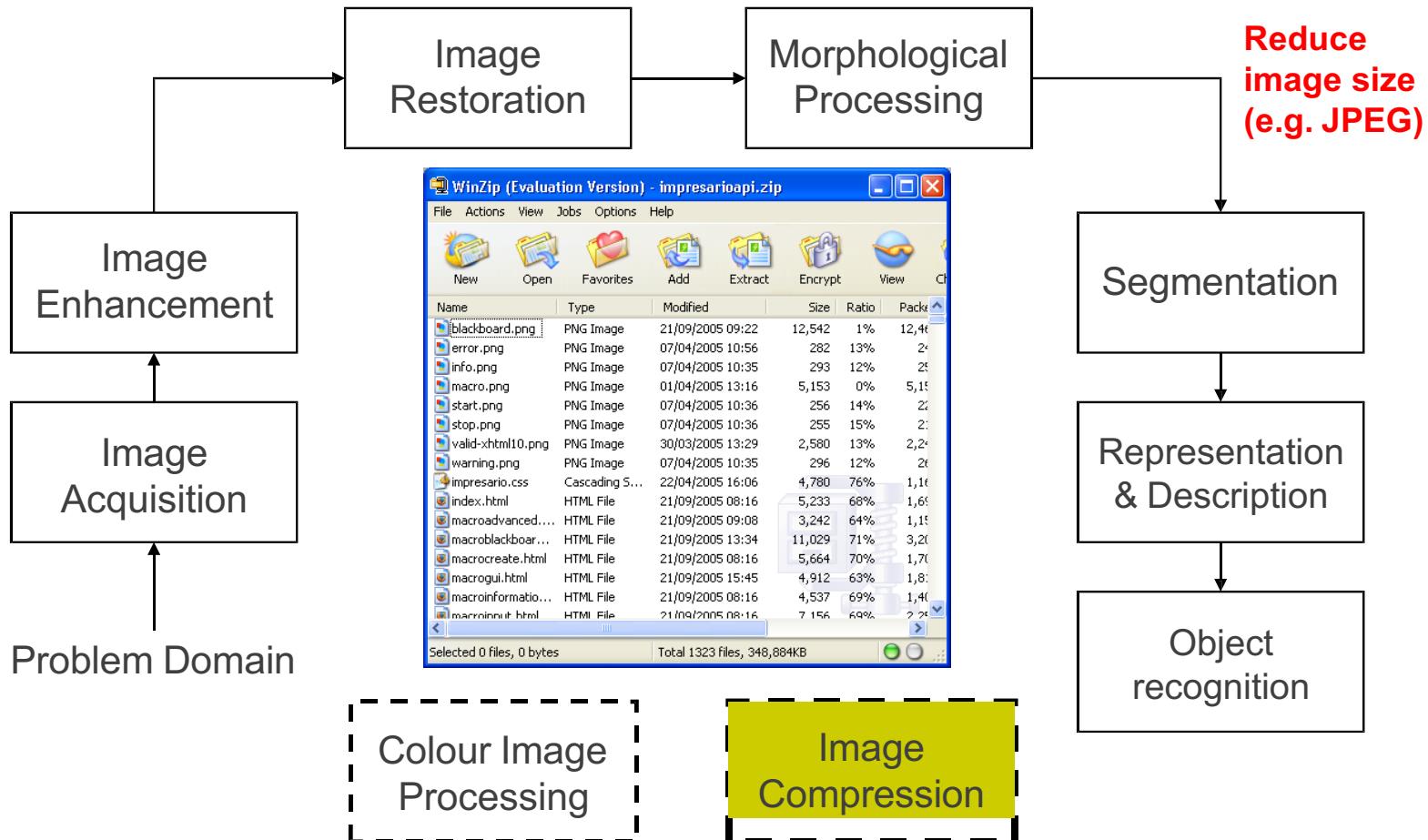


Key Stages in Digital Image Processing: Representation & Description

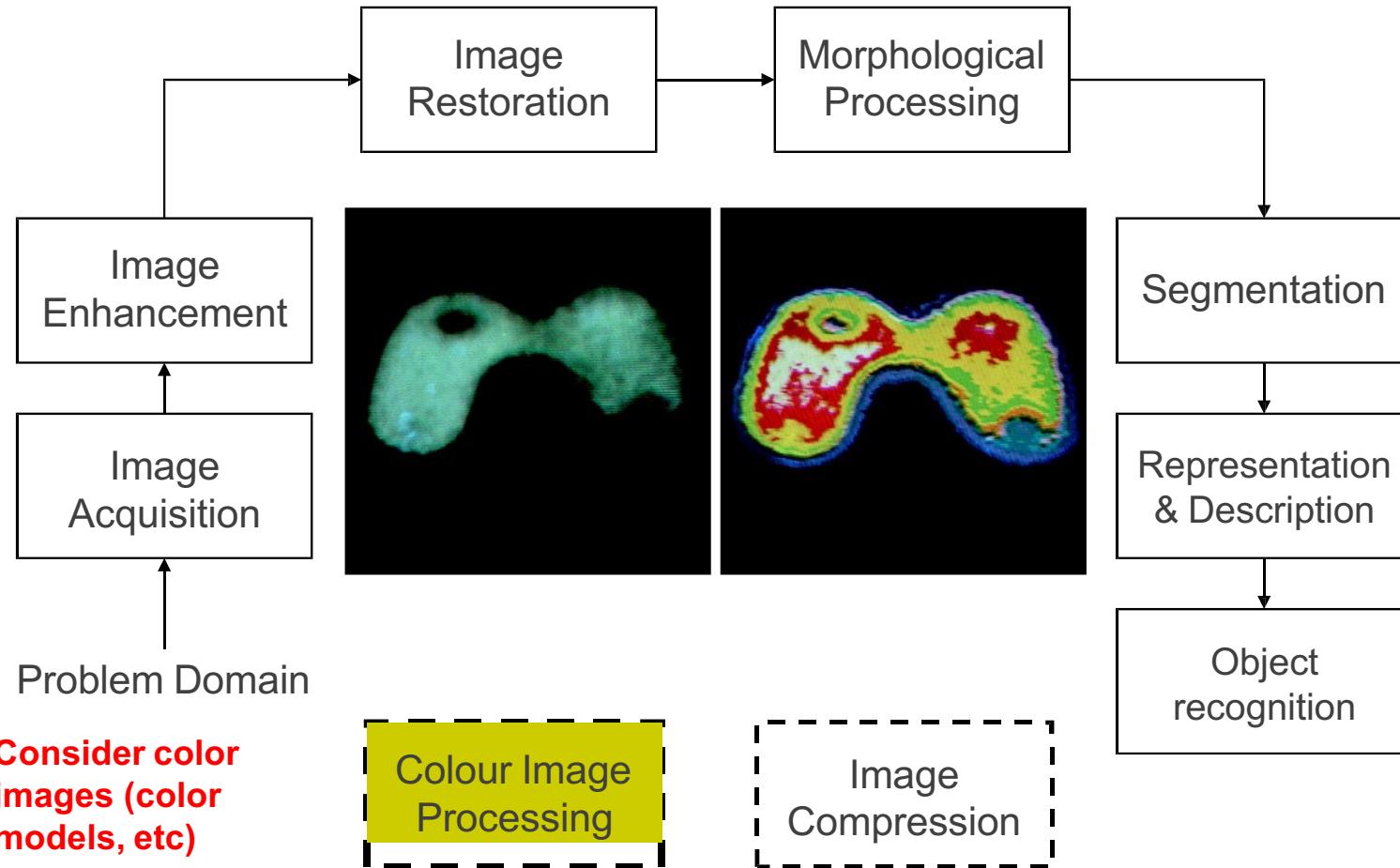
Images taken from Gonzalez & Woods, Digital Image Processing (2002)



Key Stages in Digital Image Processing: Image Compression

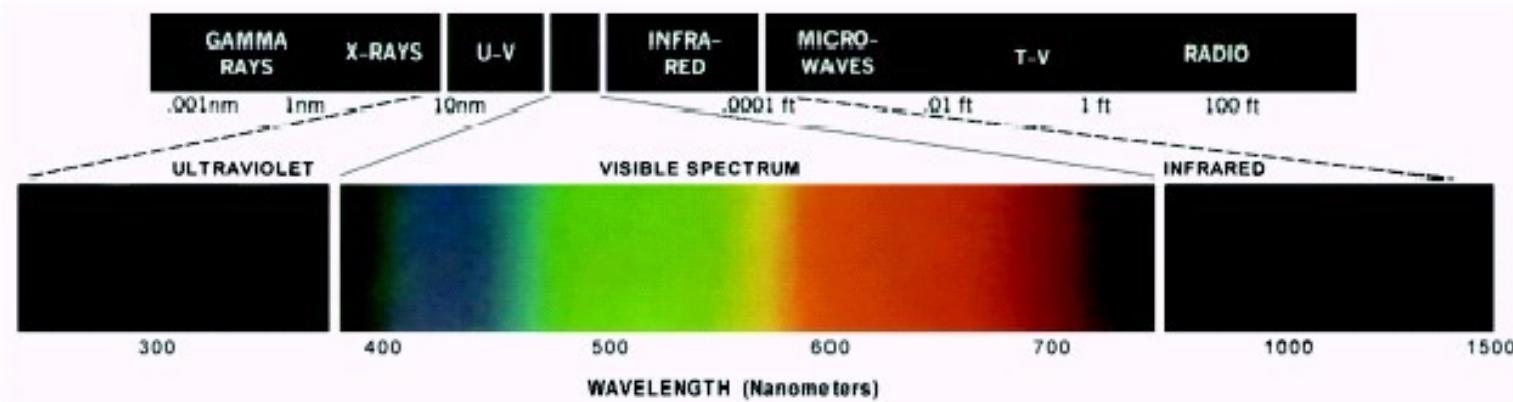


Key Stages in Digital Image Processing: Colour Image Processing



Light And The Electromagnetic Spectrum

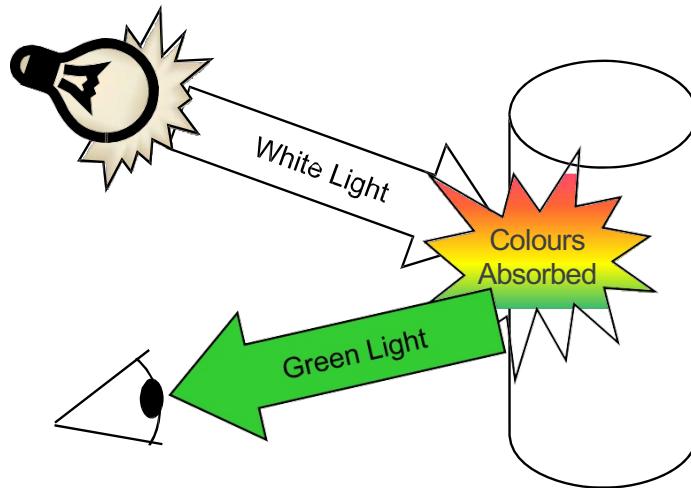
- Light: just a particular part of electromagnetic spectrum that can be sensed by the human eye
- The electromagnetic spectrum is split up according to the wavelengths of different forms of energy



Reflected Light

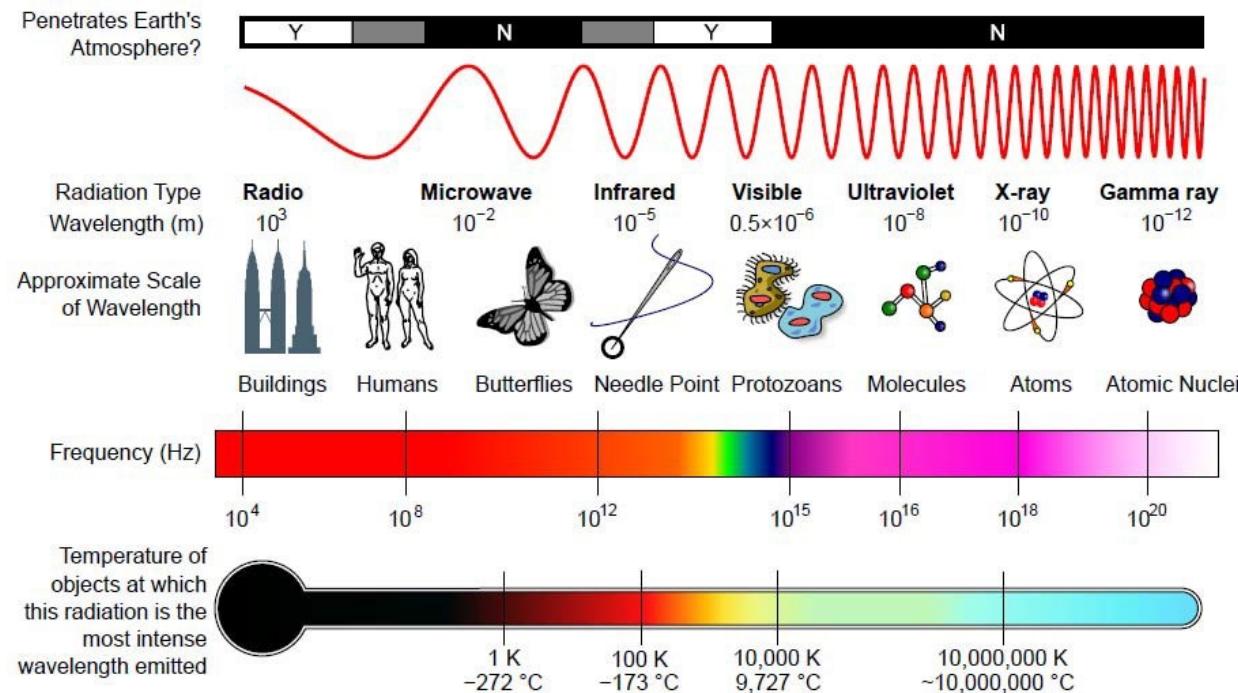
- The colours humans perceive are determined by nature of light reflected from an object
- For example, if white light (contains all wavelengths) is shone onto green object

Most wavelengths absorbed except green wavelength (color)



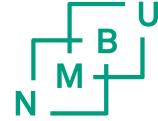
Electromagnetic Spectrum

- Images can be made from any form of EM radiation



From Wikipedia

Images from Different EM Radiation



- Radar imaging (radio waves)
- Magnetic Resonance Imaging (MRI) (Radio waves)
- Microwave imaging
- Infrared imaging
- Photographs
- Ultraviolet imaging telescopes
- X-rays and Computed tomography
- Positron emission tomography (gamma rays)
- Ultrasound (not EM waves)

Human visual system: structure of the human eye

- The lens focuses light from objects onto the retina
- Retina covered with light receptors called **cones** (6-7 million) and **rods** (75-150 million)
- Cones concentrated around fovea. Very sensitive to colour
- Rods more spread out and sensitive to low illumination levels

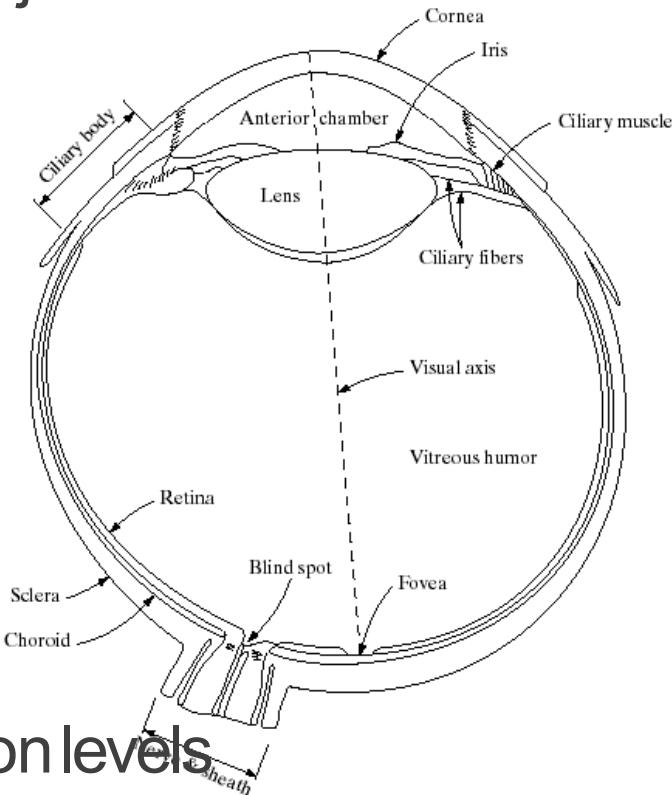
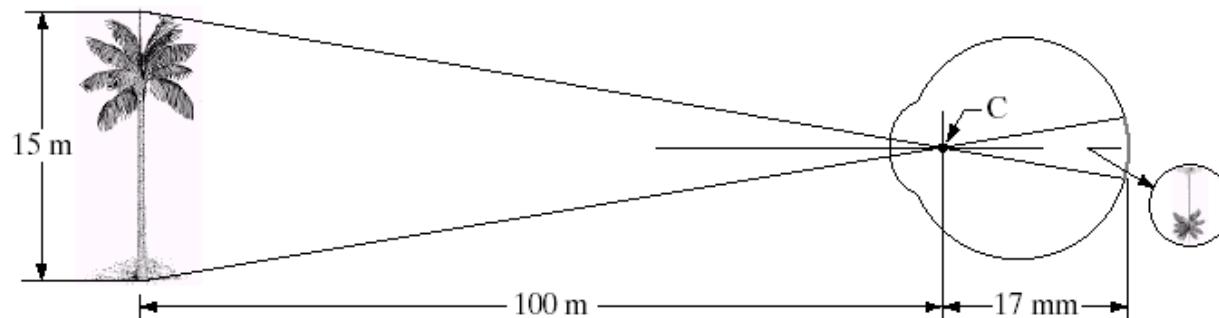
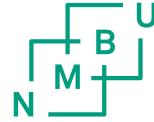


Image Formation In The Eye

- Muscles in eye can change the shape of the lens allowing us focus on near or far objects
- An image is focused onto retina exciting the rods and cones and send signals to the brain

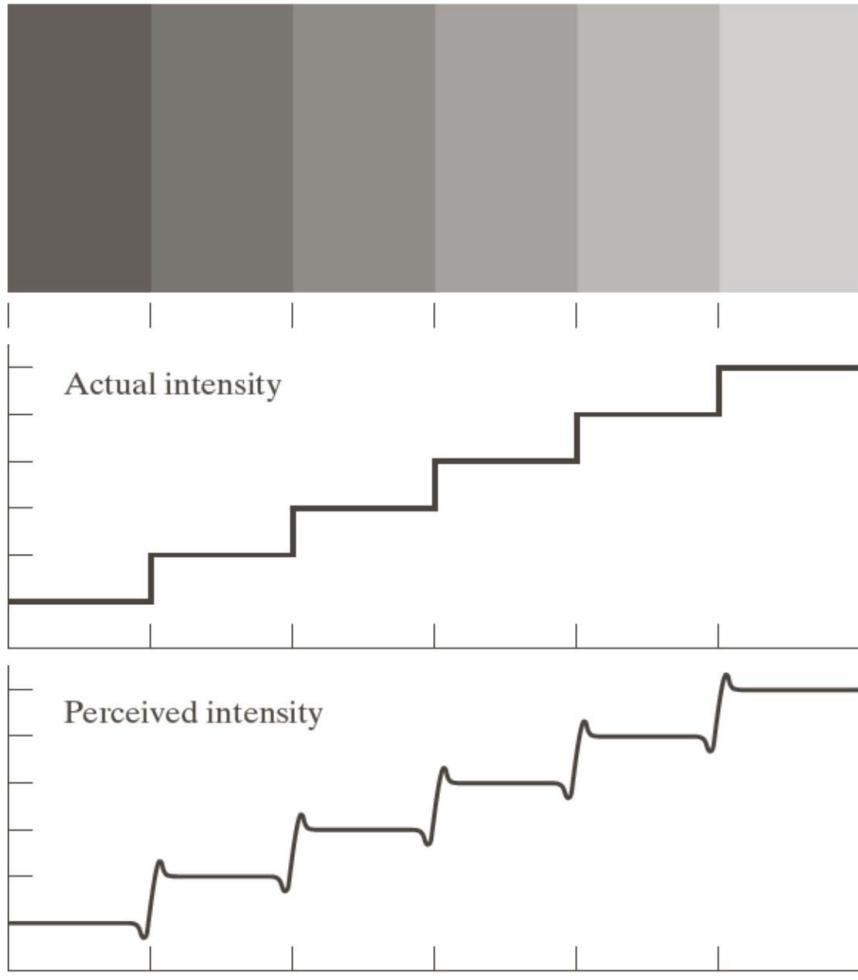




Brightness adaptation & discrimination

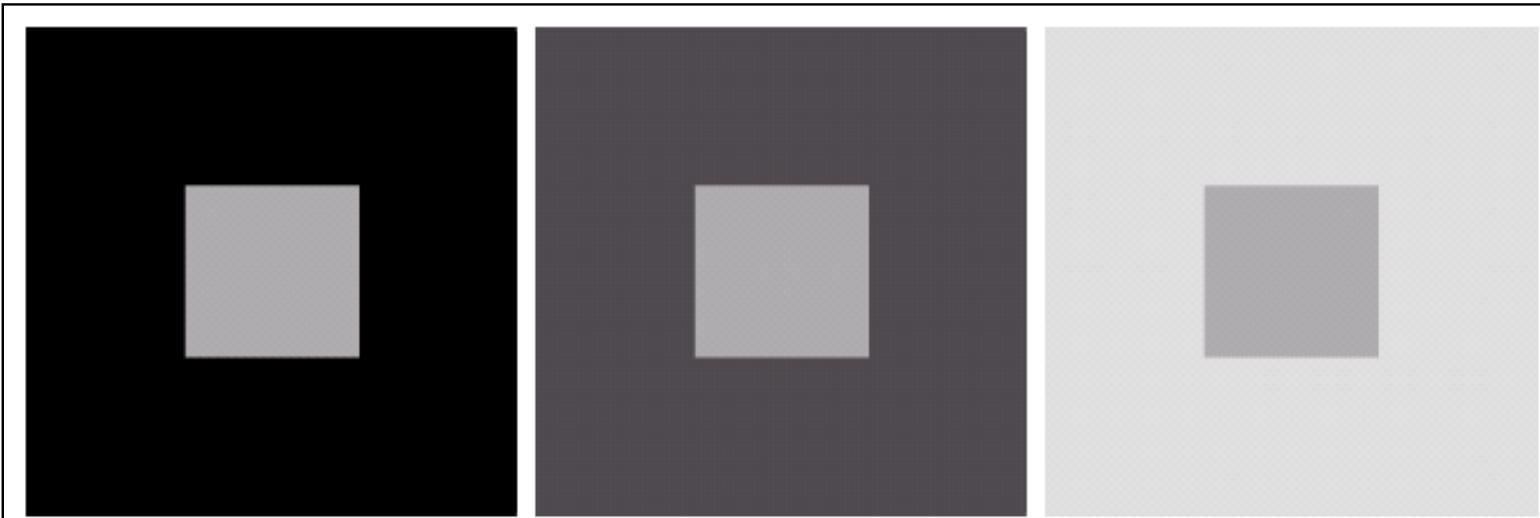
- The human visual system can perceive approximately 10^{10} different light intensity levels
- However, at any one time we can only discriminate between a much smaller number – **brightness adaptation**
- Similarly, **perceived intensity** of a region is related to the light intensities of the regions surrounding it

Brightness adaptation & discrimination

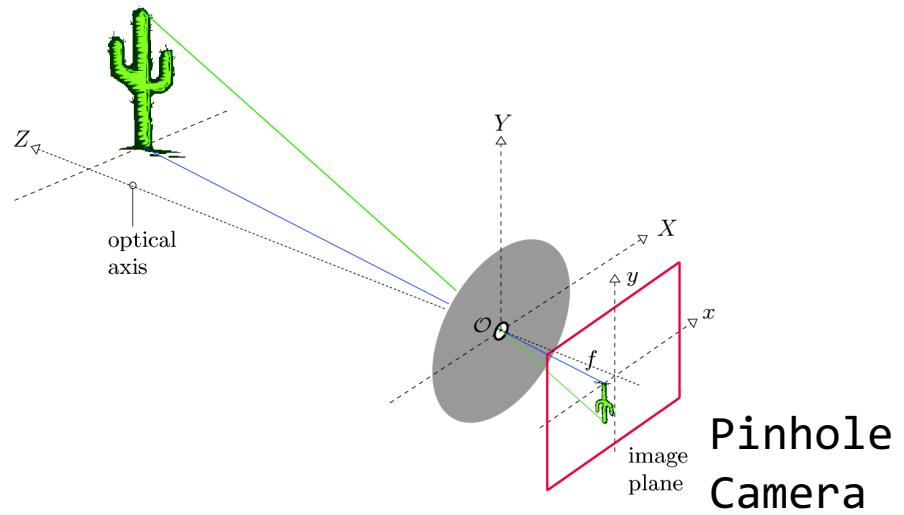
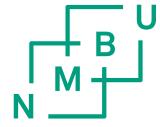


Perceived intensity
overshoots or undershoots
at areas of intensity change

Brightness adaptation & discrimination

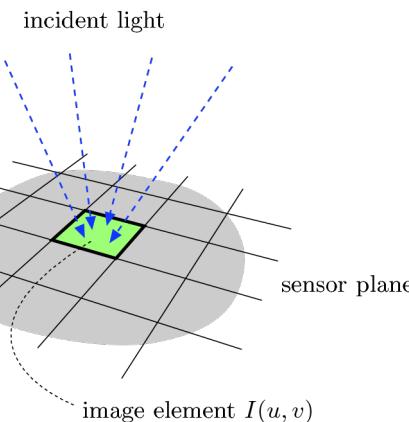
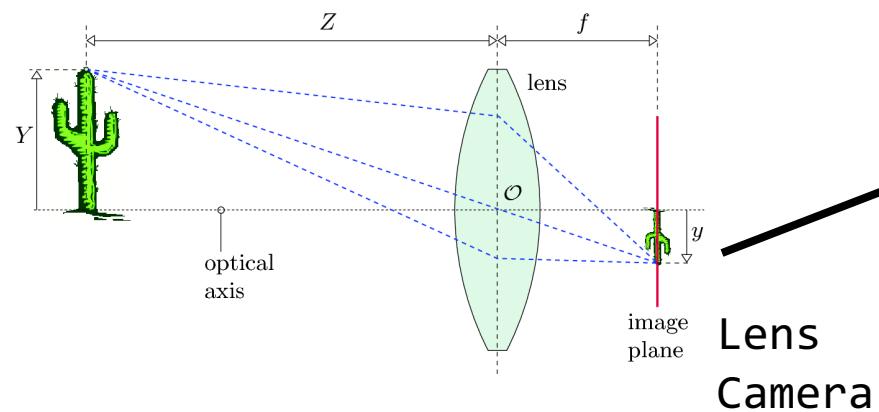


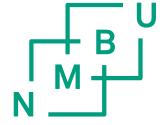
All inner squares have same intensity but appear darker as outer square (surrounding area) gets lighter



$$y = -f \frac{Y}{Z}$$

$$x = -f \frac{X}{Z}$$

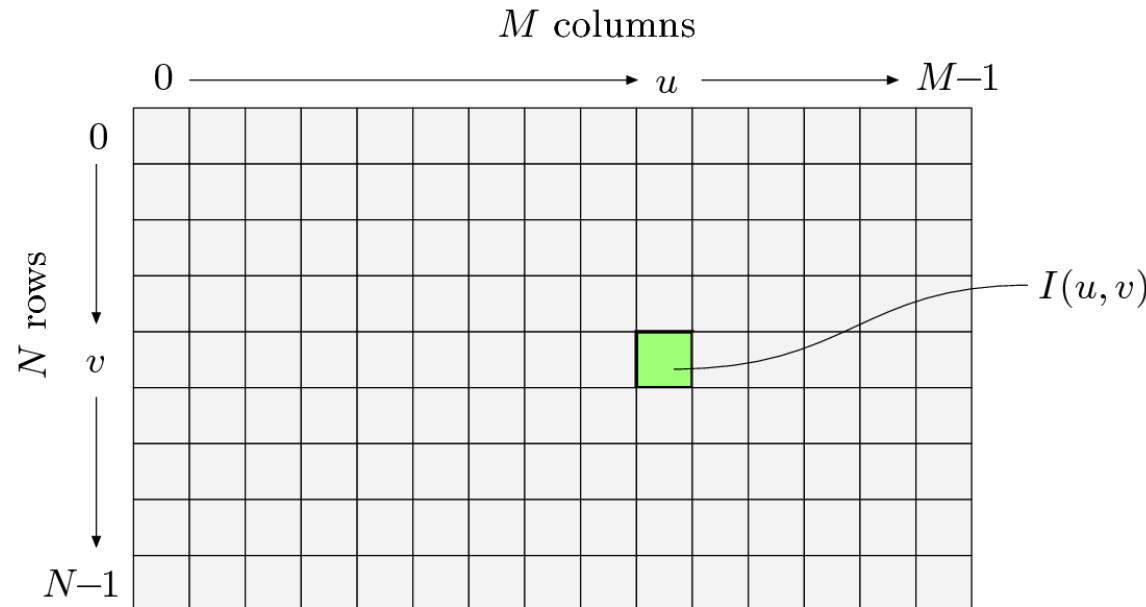




Digitalisation

- Images are represented as discrete functions and matrices

$$I(u, v) \in \mathbb{P} \quad \text{and} \quad u, v \in \mathbb{N}.$$





Representing Images

- Image data structure is 2D array of pixel values
- Pixel values are gray levels in range 0-255 or RGB colors
- Array values can be any data type (bit, byte, int, float, double, etc.)

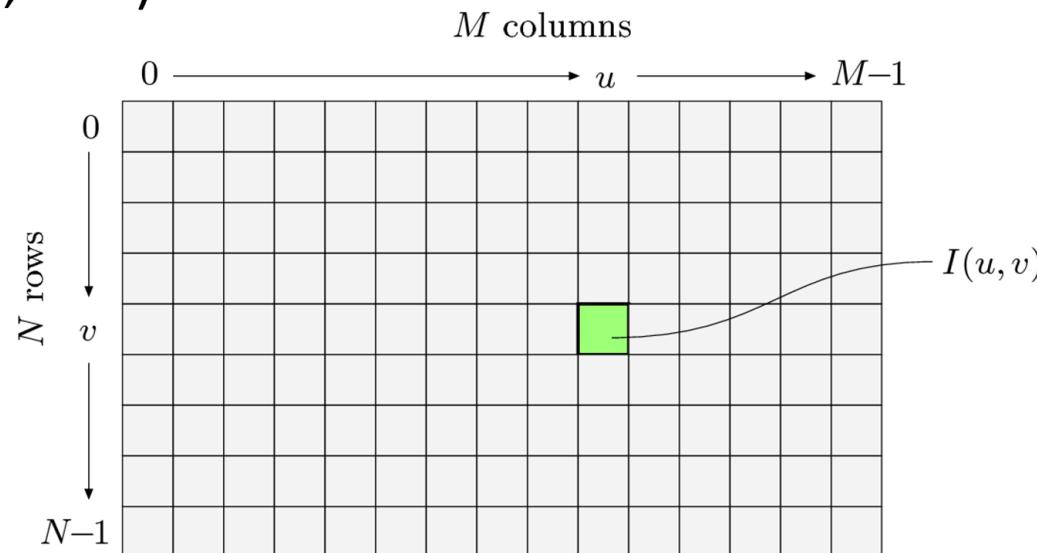
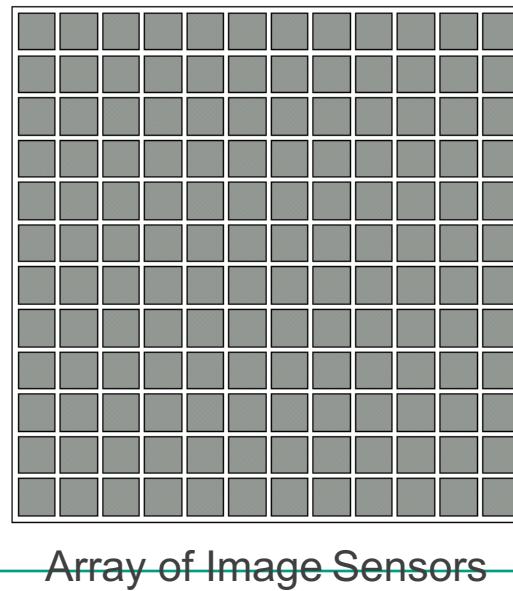
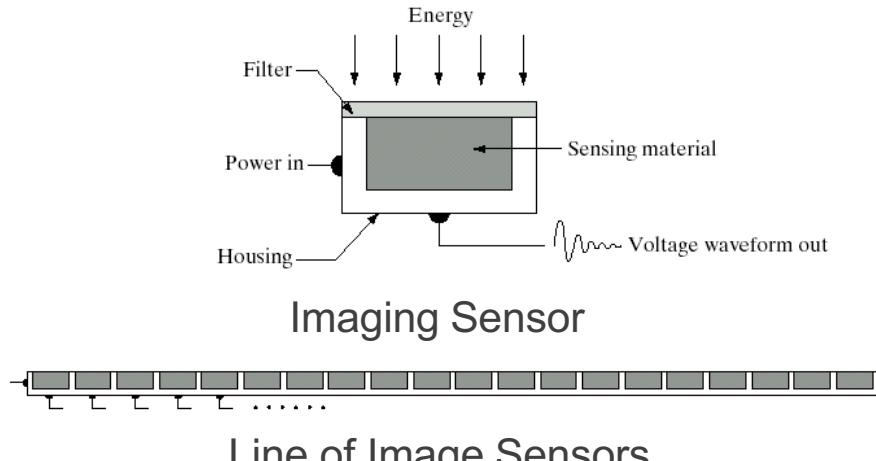
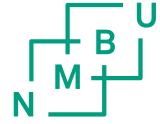


Image Sensing

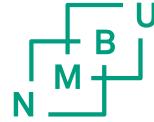
- Incoming energy (e.g. light) lands on a sensor material responsive to that type of energy, generating a voltage
- Collections of sensors are arranged to capture images





Sampling

- Spatial sampling (spatial resolution)
- Quantifications of pixel values (intensity level resolution)
- Temporal sampling (time series)



Spatial Sampling

- Cannot record image values for all (x,y)
 - Sample/record image values at discrete (x,y)
 - Sensors arranged in grid to sample image

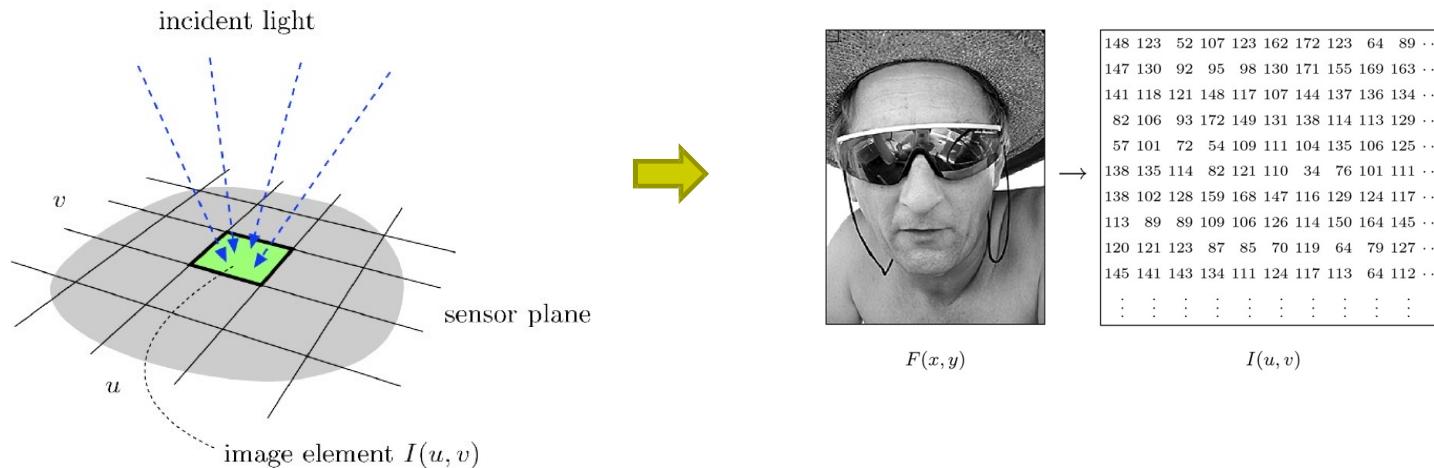


Image (Spatial) Sampling

- A digital sensor can only measure a limited number of **samples** at a **discrete** set of energy levels
 - **Sampling** can be thought of as:

Continuous signal x comb function

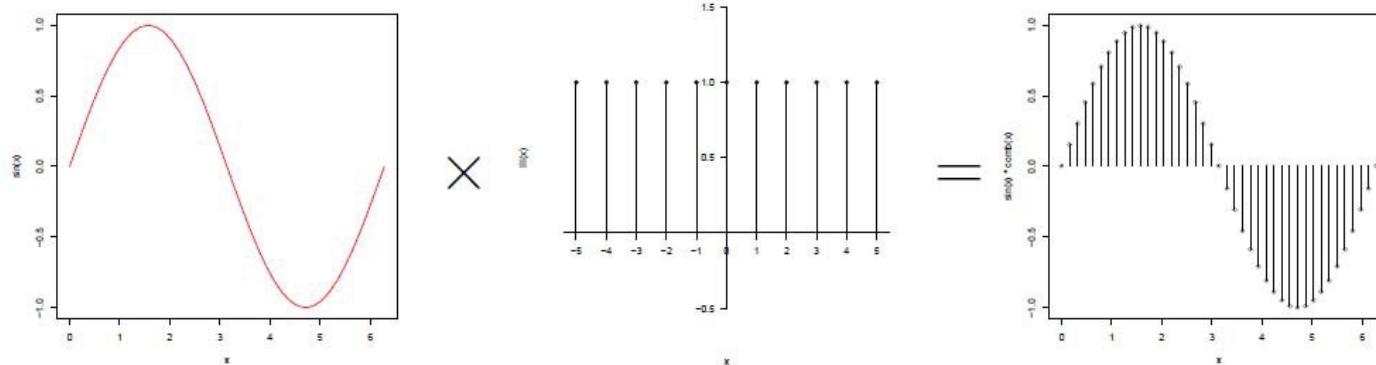


Image Quantization

- **Quantization:** process of converting continuous **analog** signal into its digital representation
- Discretize image $I(u, v)$ values
- Limit values image can take

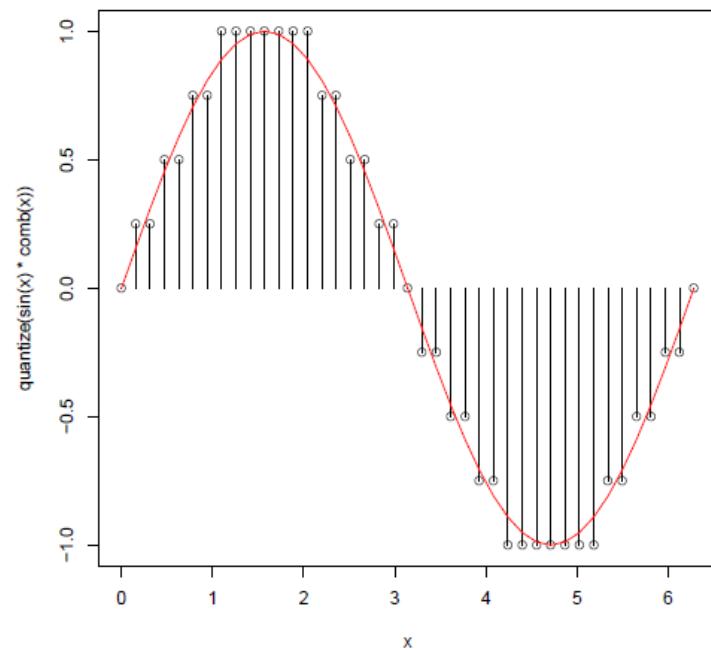
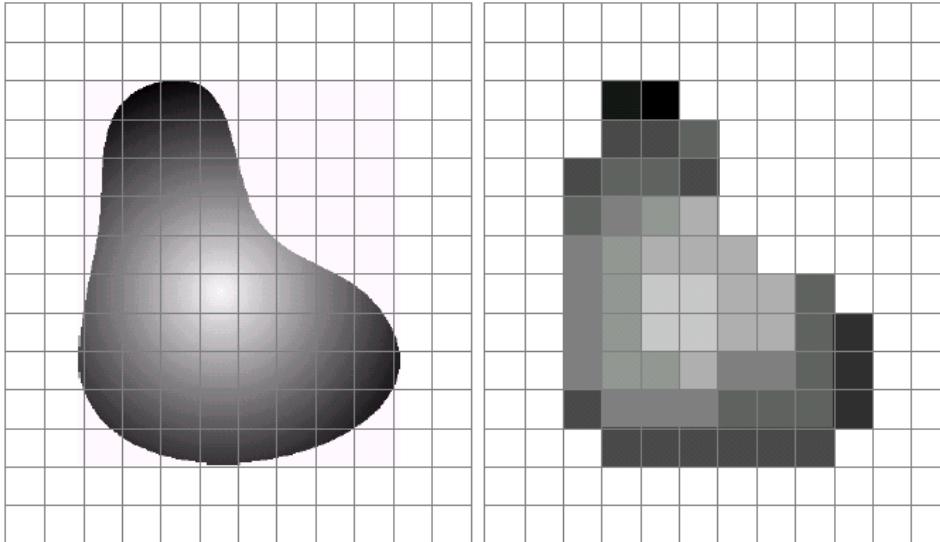


Image Sampling And Quantization



- Sampling and quantization generates **approximation** of a real world scene



Spatial Resolution

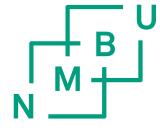
□ The ***spatial resolution*** of an image is determined by how fine/coarse sampling was carried out

□ **Spatial resolution:** smallest discernable image detail

- Vision specialists talk about image resolution
- Graphic designers talk about *dots per inch* (DPI)

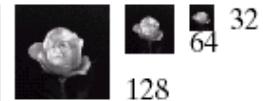


KenRockwell.com



Spatial Resolution

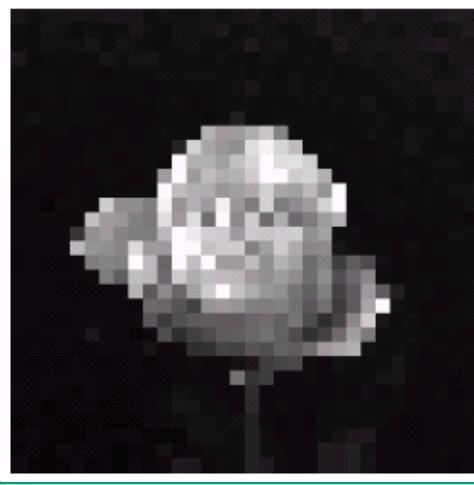
Images taken from Gonzalez & Woods, Digital Image Processing (2002)



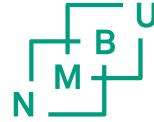
Spatial Resolution: Stretched Images

N M B U

Images taken from Gonzalez & Woods, Digital Image Processing (2002)



Resolution: How Much Is Enough?



□ **Example:** Picture on right okay for counting number of cars, but not for reading the number plate

Intensity Level Resolution



Low Detail



Medium Detail



High Detail

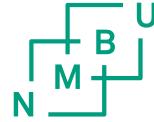


Resolution: How Much Is Enough?

□ The big question with resolution is always *how much is enough?*

- Depends on what is in the image (*details*) and what you would like to do with it (*applications*)
- Key questions:
 - Does image look aesthetically pleasing?
 - Can you see what you need to see in image?



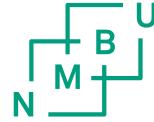


Intensity Level Resolution

□ **Intensity level resolution:** number of intensity levels used to represent the image

- The more intensity levels used, the finer the level of detail discernable in an image
- Intensity level resolution usually given in terms of number of bits used to store each intensity level

| Number of Bits | Number of Intensity Levels | Examples |
|----------------|----------------------------|--------------------|
| 1 | 2 | 0, 1 |
| 2 | 4 | 00, 01, 10, 11 |
| 4 | 16 | 0000, 0101, 1111 |
| 8 | 256 | 00110011, 01010101 |
| 16 | 65,536 | 1010101010101010 |



Intensity Level Resolution

□ **Intensity level resolution:** number of intensity levels used to represent the image

- The more intensity levels used, the finer the level of detail discernable in an image
- Intensity level resolution usually given in terms of number of bits used to store each intensity level

| Number of Bits | Number of Intensity Levels | Examples |
|----------------|--|--------------------|
| | $\text{Int_levels} = 2^{(\text{bits})}$ | 0, 1 |
| | | 00, 01, 10, 11 |
| 4 | 16 | 0000, 0101, 1111 |
| 8 | 256 | 00110011, 01010101 |
| 16 | 65,536 | 1010101010101010 |

Intensity Level Resolution

256 grey levels (8 bits per pixel)



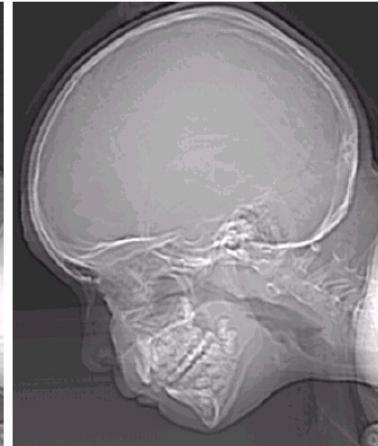
128 grey levels (7 bpp)



64 grey levels (6 bpp)



32 grey levels (5 bpp)



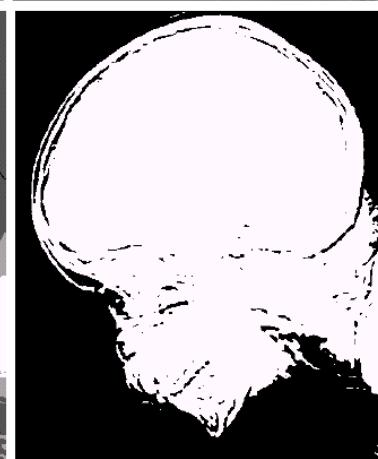
16 grey levels (4 bpp)



8 grey levels (3 bpp)



4 grey levels (2 bpp)

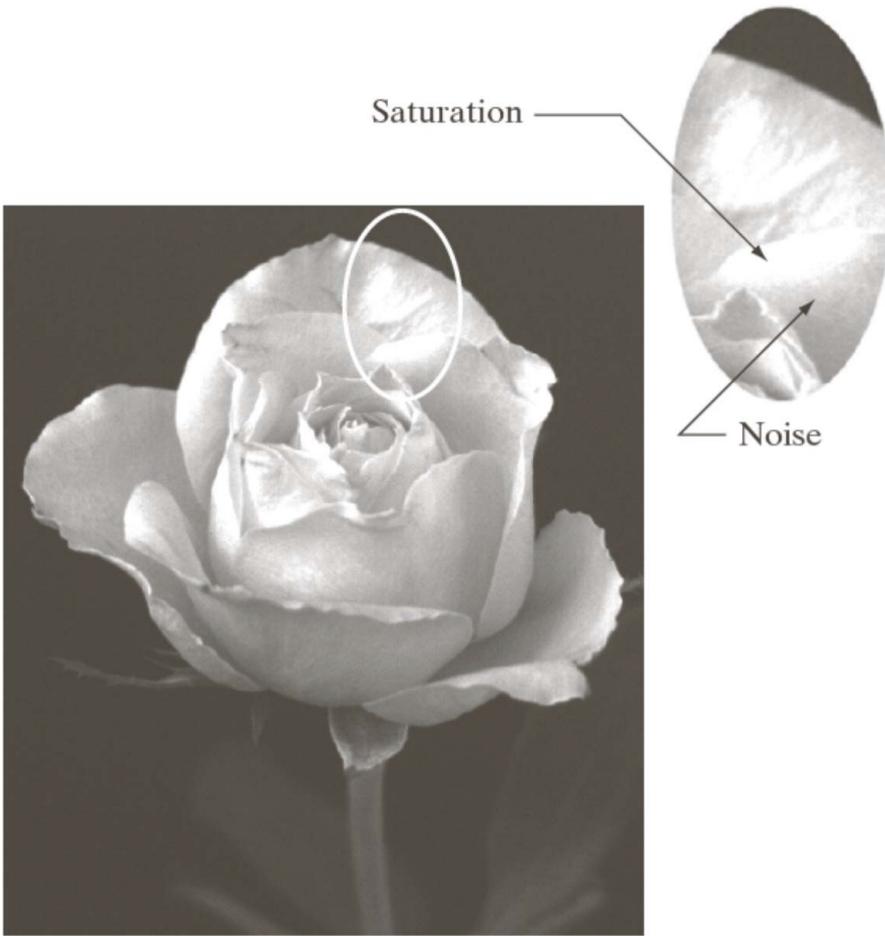


2 grey levels (1 bpp)



Saturation & Noise

Images taken from Gonzalez & Woods, Digital Image Processing (2002)



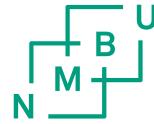
Saturation: highest intensity value above which color is washed out

Noise: grainy texture pattern

Number of gray levels



Figure 1.25 Four representations of the same image, with variation in the number of gray levels used. From the upper left: 32; 16; 8; 4. In all cases, a full 256×256 array of pixels is retained. Each step in the coarsening of the image is accomplished by rounding the brightness of the original pixel value.



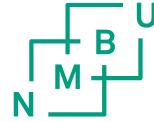
Bits & Bytes

Everything in a computer is 0's and 1's ... what does that mean? The **bit** stores just a 0 or 1 .. it's the smallest building block of storage.

Byte

- One byte = grouping of 8 bits
- e.g. 0 1 0 1 1 0 1 0
- One byte can store one letter, e.g. 'A' or 'x'

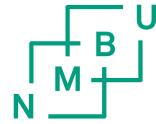
<https://web.stanford.edu/class/cs101/bits-bytes.html>



Bits & Bytes

- In general: add 1 bit, double the number of patterns
- 1 bit - 2 patterns
- 2 bits - 4
- 3 bits - 8
- 4 bits - 16
- 5 bits - 32
- 6 bits - 64
- 7 bits - 128
- 8 bits - 256
- Mathematically: n bits yields 2^n patterns (2 to the n th power)

<https://web.stanford.edu/class/cs101/bits-bytes.html>



Bits & Bytes

One Byte - 256 Patterns

- Need to know:
- 1 byte is group of 8 bits
- 8 bits can make 256 different patterns
- How to use the 256 patterns?
- How to store a number in a byte?
- Start with 0, go up, one pattern per number, until run out of patterns
- 0, 1, 2, 3, 4, 5, ... 254, 255
- One byte holds a number 0..255
- i.e. with 256 distinct patterns, we can store a number in the range 0..255
- Code: `pixel.setRed(n)` took a number 0..255. Why?
- The red/green/blue image numbers are each stored in **one byte**

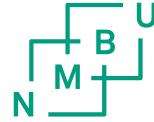
<https://web.stanford.edu/class/cs101/bits-bytes.html>

Bits & Bytes

Closest 1000 is
 $2^{10} = 1024$
 $1kb = 1024$ bytes

Bytes

- "Byte" - unit of information storage
- A document, an image, a movie .. how many bytes?
- 1 byte is enough to hold 1 typed letter, e.g. 'b' or 'X'
- Later we'll look at storage in: RAM, hard drives, flash drives
- All measured in bytes, despite being very different hardware
- **Kilobyte**, KB, about 1 thousand bytes
- **Megabyte**, MB, about 1 million bytes
- **Gigabyte**, GB, about 1 billion bytes
- **Terabyte**, TB, about 1 trillion bytes (rare)



Digitalisation of images

Grayscale (Intensity Images):

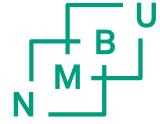
| Chan. | Bits/Pix. | Range | Use |
|-------|-----------|-----------|---|
| 1 | 1 | 0...1 | Binary image: document, illustration, fax |
| 1 | 8 | 0...255 | Universal: photo, scan, print |
| 1 | 12 | 0...4095 | High quality: photo, scan, print |
| 1 | 14 | 0...16383 | Professional: photo, scan, print |
| 1 | 16 | 0...65535 | Highest quality: medicine, astronomy |

Color Images:

| Chan. | Bits/Pix. | Range | Use |
|-------|-----------|-----------------|---------------------------------------|
| 3 | 24 | $[0...255]^3$ | RGB, universal: photo, scan, print |
| 3 | 36 | $[0...4095]^3$ | RGB, high quality: photo, scan, print |
| 3 | 42 | $[0...16383]^3$ | RGB, professional: photo, scan, print |
| 4 | 32 | $[0...255]^4$ | CMYK, digital prepress |

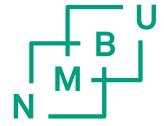
Special Images:

| Chan. | Bits/Pix. | Range | Use |
|-------|-----------|--------------------------|--|
| 1 | 16 | $-32768...32767$ | Whole numbers pos./neg., increased range |
| 1 | 32 | $\pm 3.4 \cdot 10^{38}$ | Floating point: medicine, astronomy |
| 1 | 64 | $\pm 1.8 \cdot 10^{308}$ | Floating point: internal processing |



Sampling

- Spatial sampling (spatial resolution)
- Quantifications of pixel values (intensity level resolution)
- Temporal sampling (time series)



Examples of image processing

Example Operation: Noise Removal

Noisy Image



Denoised Image

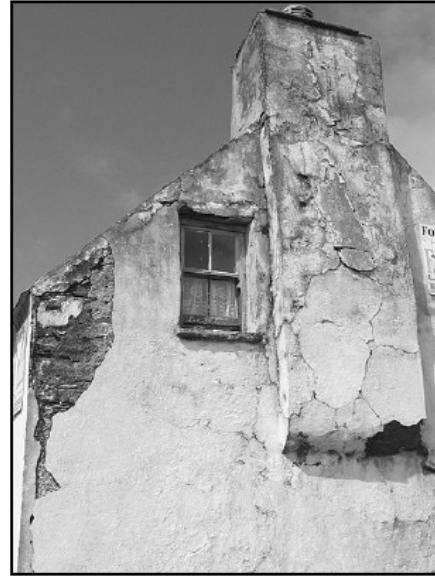


Think of noise as white specks on a picture (random or non-random)

Example: Contrast Adjustment



Low Contrast



Original Contrast



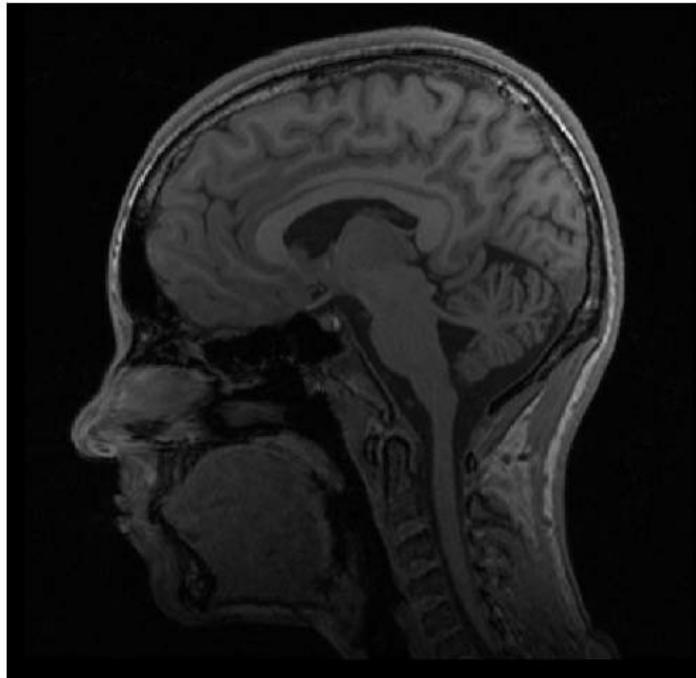
High Contrast

Example: Edge Detection



Applications of Image Processing/Analysis

Medicine



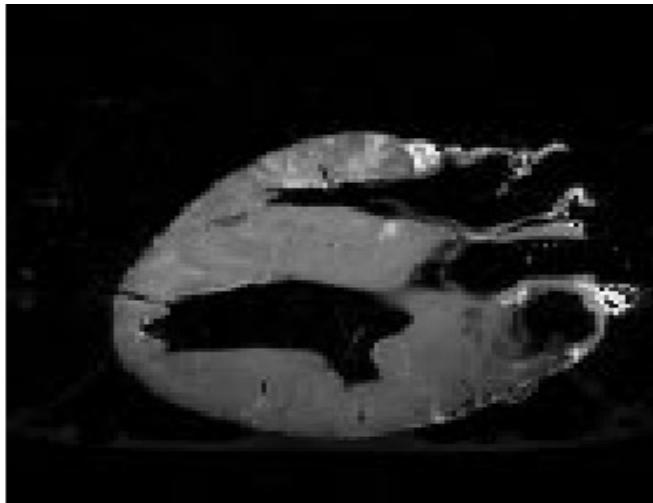
Credit: Dr. Janet Lainhart, UofU Psychiatry

Security, Biometrics

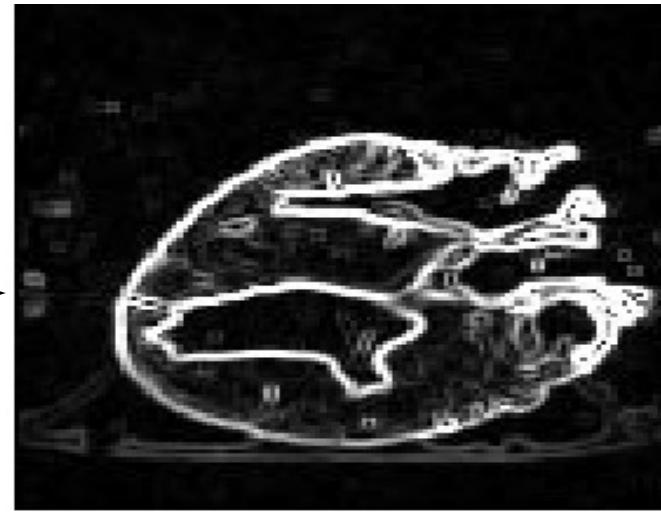


Applications of Image Processing: Medicine

Images taken from Gonzalez & Woods, Digital Image Processing (2002)



Original MRI Image of a Dog Heart



Edge Detection Image

Applications of Image Processing

Satellite Imagery



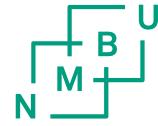
Credit: NASA

Personal Photos

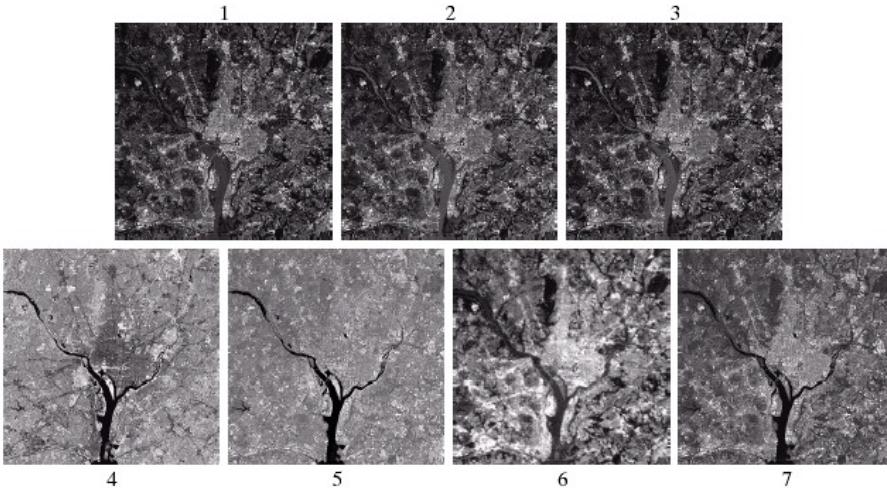


Credit: Tom Fletcher

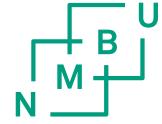
Applications of Image Processing: Geographic Information Systems (GIS)



- Terrain classification
- Meteorology (weather)

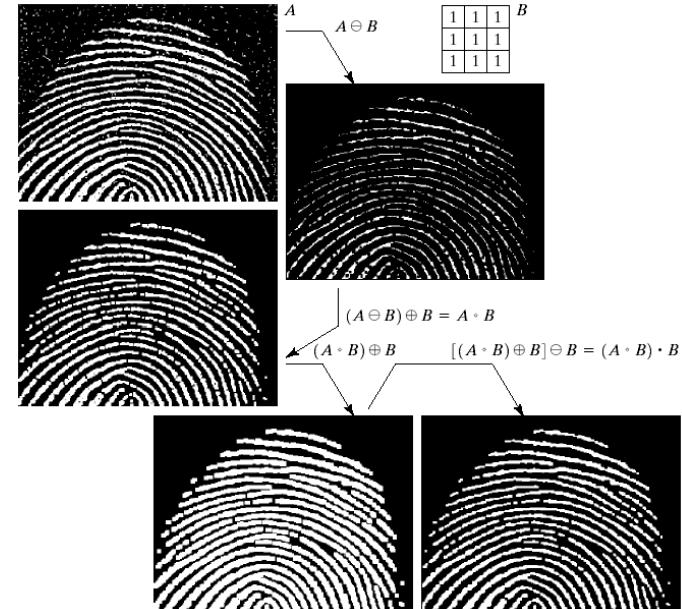
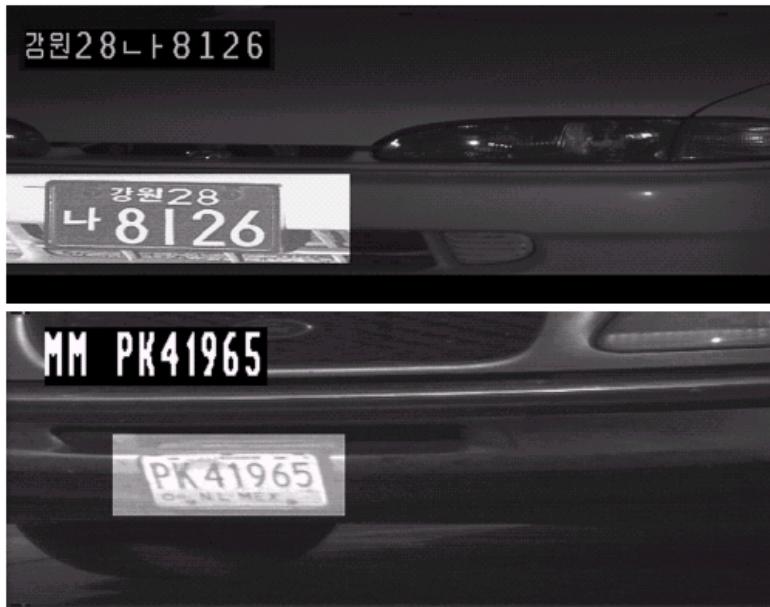


Applications of Image Processing: Law Enforcement



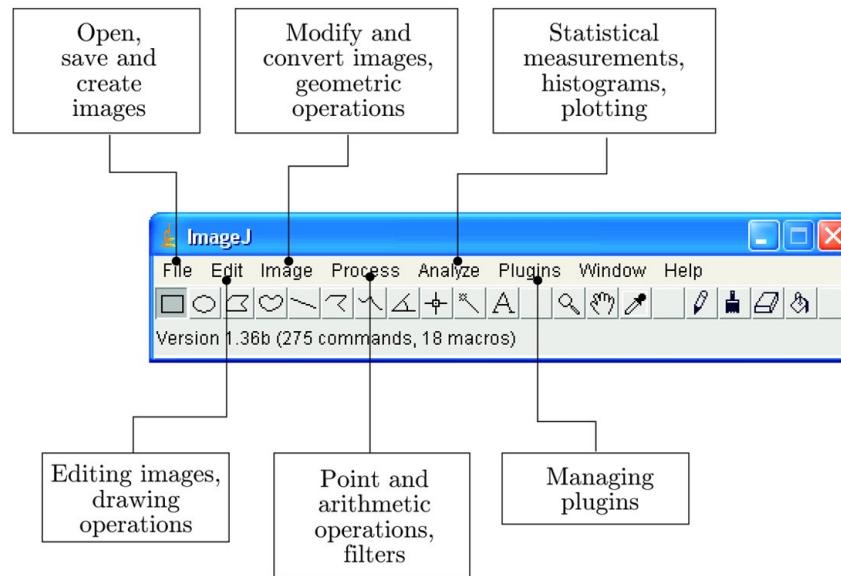
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

- Number plate recognition for speed cameras or automated toll systems
- Fingerprint recognition

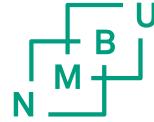


ImageJ/Fiji

- Open source Java Image processing software
- Developed by Wayne Rasband at Nat. Inst for Health (NIH)
 - Many image processing algorithms **already implemented**
 - New image processing algorithms can also be implemented easily
 - Nice click-and-drag interface

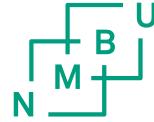


Wayne Rasband (right)



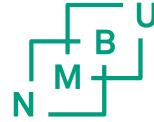
ImageJ: Key Features

- **Interactive tools** for image processing of images
 - Supports many image file formats (JPEG, PNG, GIF, TIFF, BMP, DICOM, FITS)
- **Plug-in mechanism** for implementing new functionality, extending ImageJ
- **Macro language + interpreter**: Easy to implement large blocks from small pieces without knowing Java



Useful image formats

- JPEG
 - Destructive compression
- TIFF (Tagged Image File Format)
 - Non destructive compression
 - Used in ImageJ for storage of images and stacks of images
- GIF
 - Graphics Interchange Format
 - Used much on internet. Transparency, animated images
- PNG (Portable Network Format)
 - A good alternative to JPEG.
 - Non destructive
 - Transparency
- Raw images
 - Digital photo «negative film»
 - RAW (Canon)
 - NEF (Nikon)

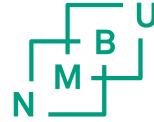


Formats for image analysis

- Record in JPEG or RAW
- Convert to TIFF or PNG
- In between savings in TIFF or PNG

Nice video on JPEG compression:

https://www.youtube.com/watch?v=Ba89cI9eIg8&ab_channel=LeolIsikdogan



JPEG

- JPEG (or JPG) is an electronic image format developed by Joint Photographic Experts Group
- Uses lossy compression techniques
- The files end up smaller than with other non lossy compression techniques
- JPEG is often used to store digital images, especially images on the internet

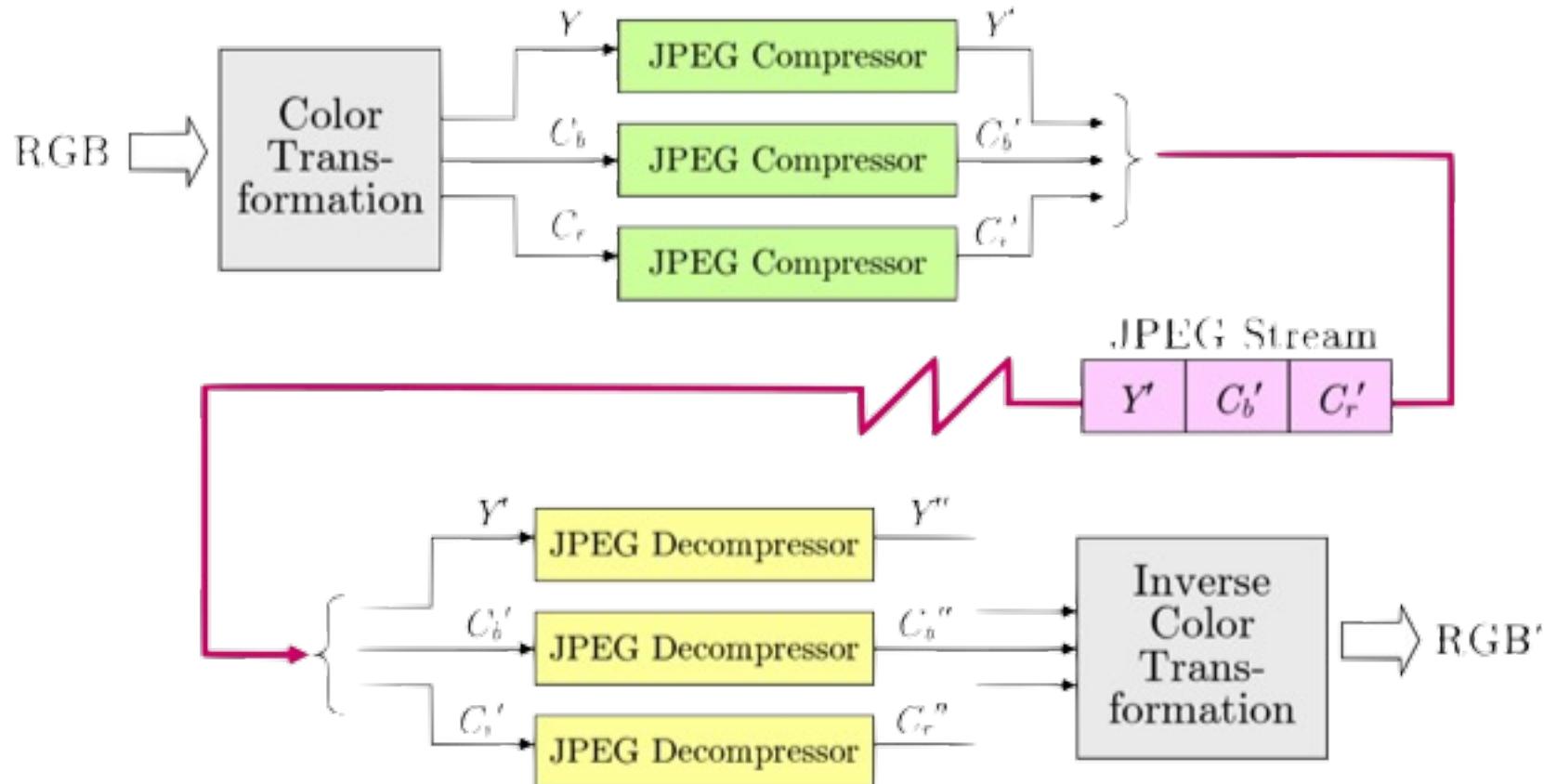


JPEG

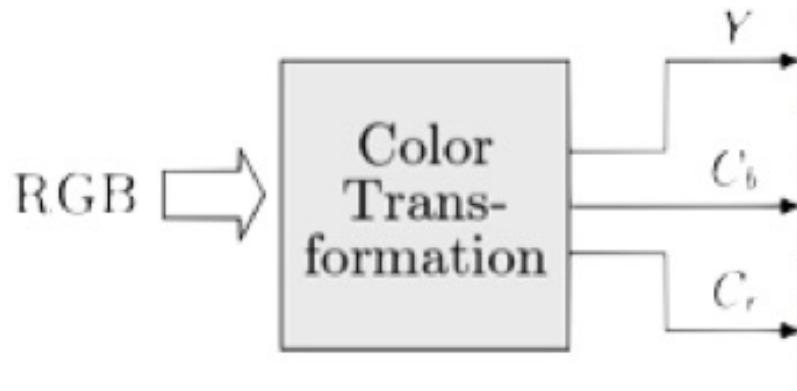
- Converting colours and downsampling
 - The human eye is not so sensitive to colour changes as to intensity variations
 - The colour component is more compressed than the intensity component
 - The compression is carried out to simulate the "reality" using COSINE transformations
 - Higher frequencies are not important for the visual perception

JPEG algorithm

N M B U



JPEG algorithm part 1



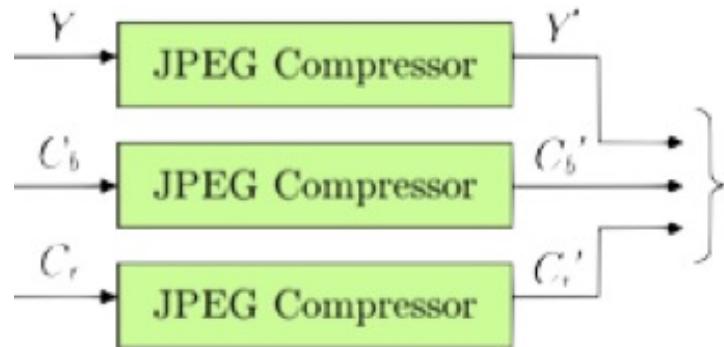
- Colour transformation from RGB to YC_bC_r

RGB = red, green, blue

Y=lightness

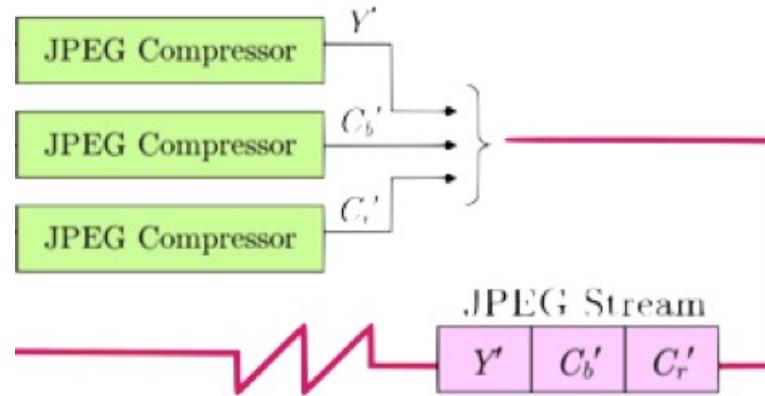
C_b,C_r = colour channels blue and red

JPEG algorithm part 2



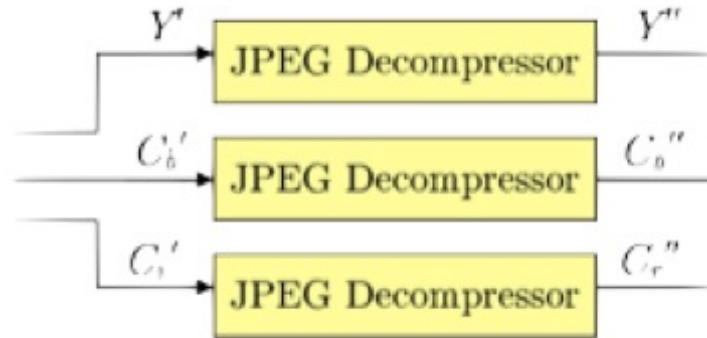
- YC_bC_r is compressed via cosine transformation
 - Coefficients (frequencies) are produced in the transformation
 - High frequencies (sharpness) are removed
 - C_b and C_r are treated differently
 - Humans separate red/green better than blue/yellow

JPEG algorithm part 3



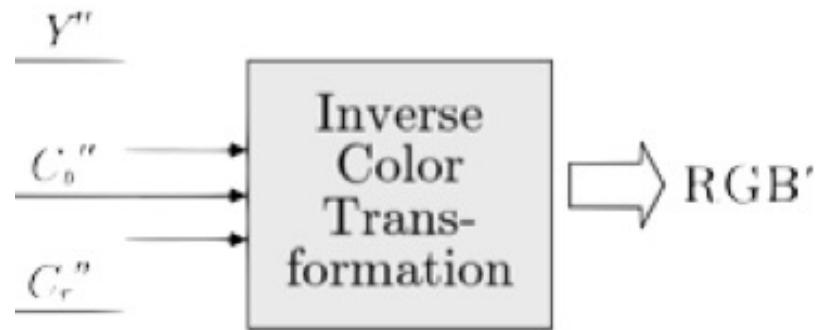
- The remaining table of coefficients are compressed without loss (JPEG stream)

JPEG algorithm part 4

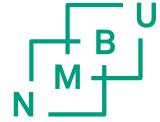


- Decompression is done with the inverse cosine transformation

JPEG algorithm part 5

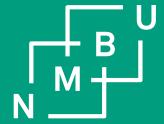


- $Y''C_b''C_r''$ are converted to RGB'
- RGB' is unpacked different from RGB
- RGB' comes with a factor of "Quality"



JPEG and “artifacts”

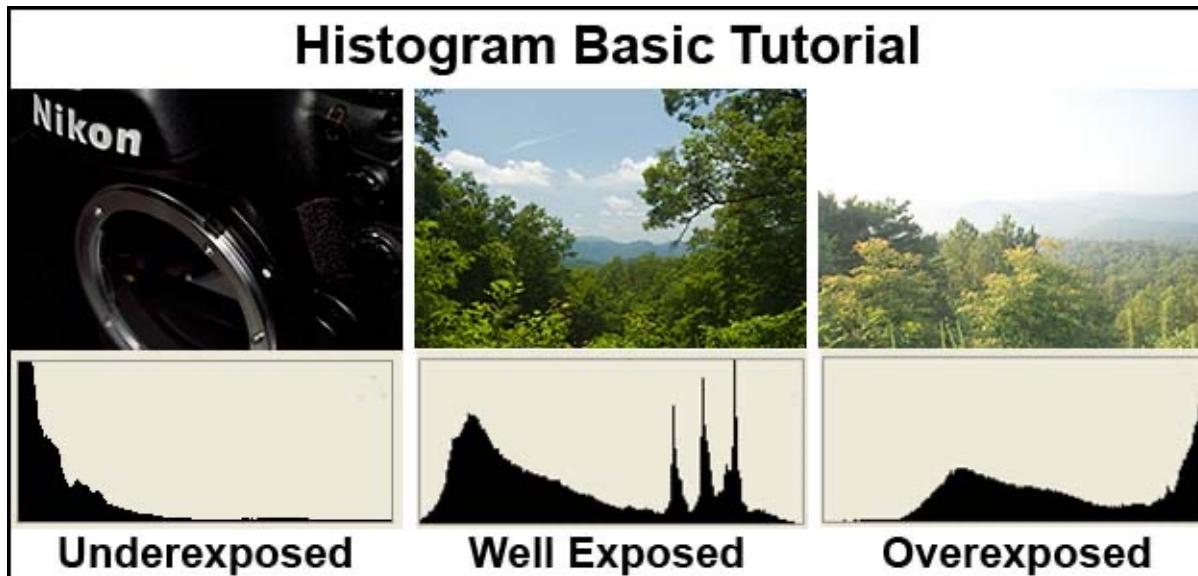
- Quality of the compression: Q_{jpeg}
- Digital cameras use JPEG with a defined compression
- JPEG should not be used when storing images during processing and analysis
- Use TIF, PNG



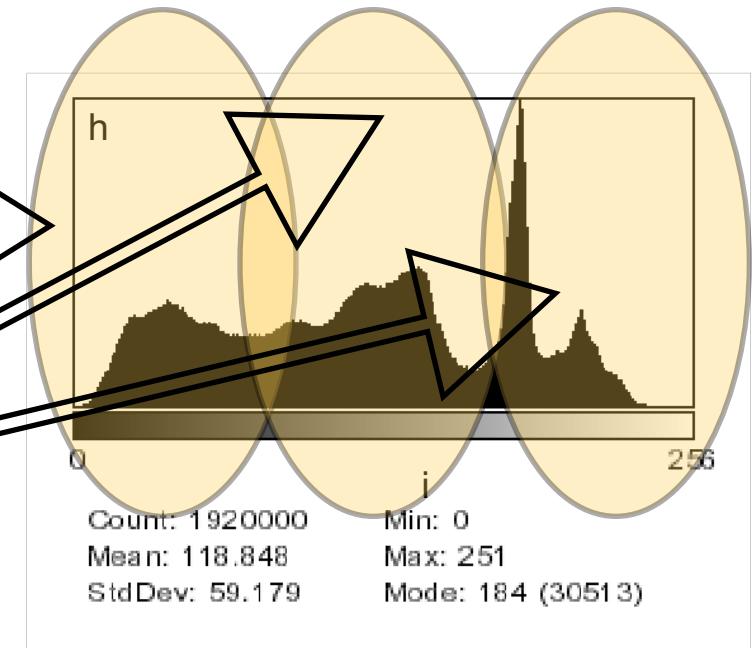
INF250

2: «Histograms and Image Statistics»

What is a histogram?



What is a histogram?



$h(i)$ = the *number* of pixels in I with the intensity value i

What is a histogram?

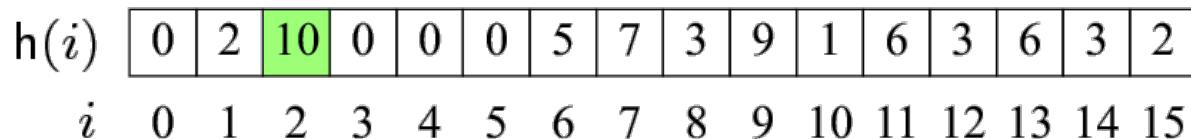
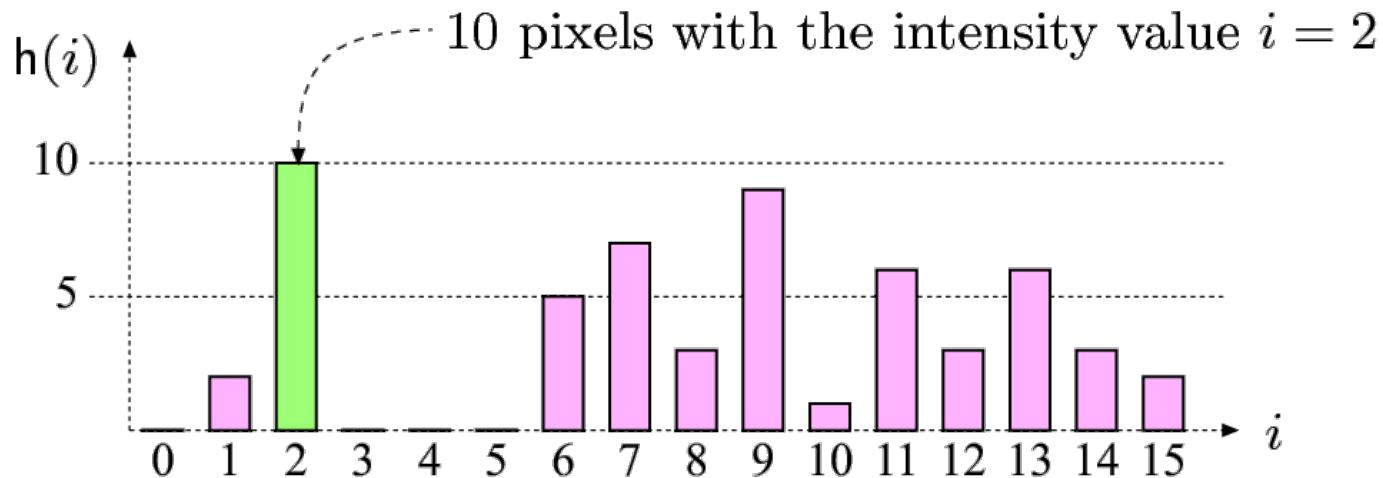
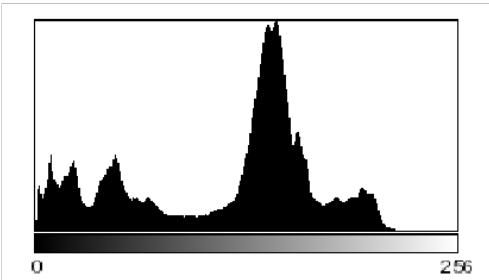
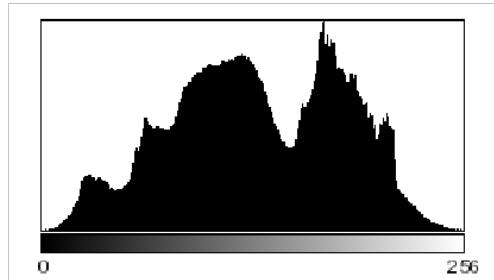


Figure 3.3 Histogram vector for an image with $K = 16$ possible intensity values. The indices of the vector element $i = 0 \dots 15$ represent intensity values. The value of 10 at index 2 means that the image contains 10 pixels of intensity value 2.

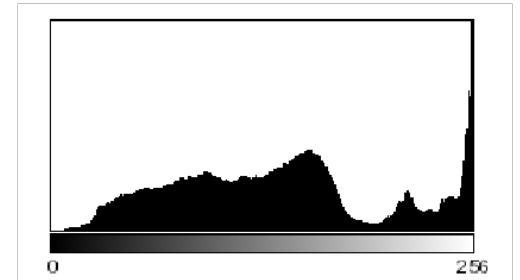
Correct exposure?



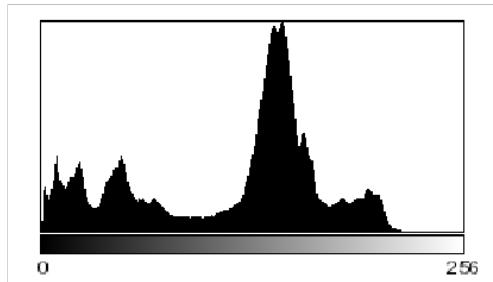
(a)



(b)

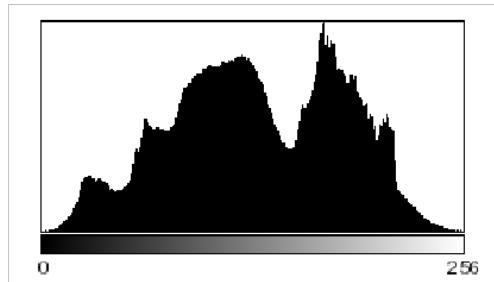


(c)



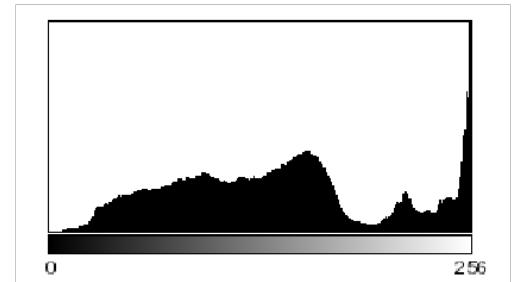
(a)

Under exposed



(b)

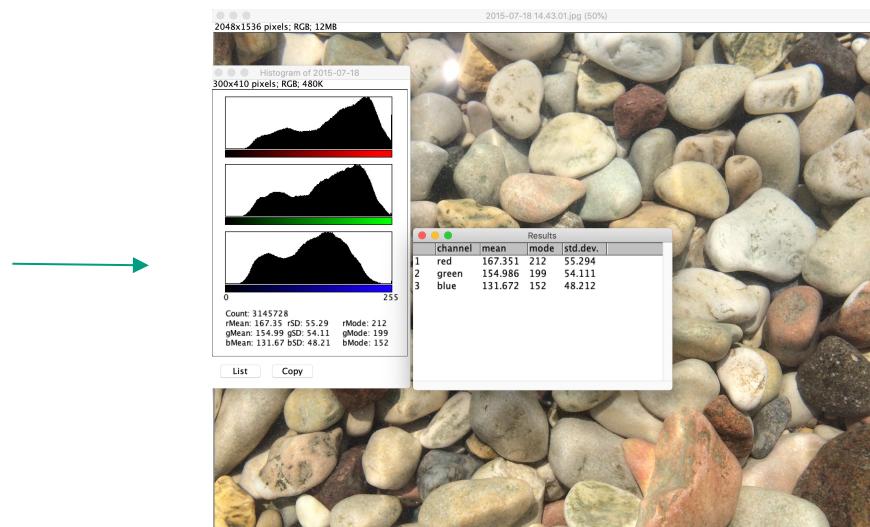
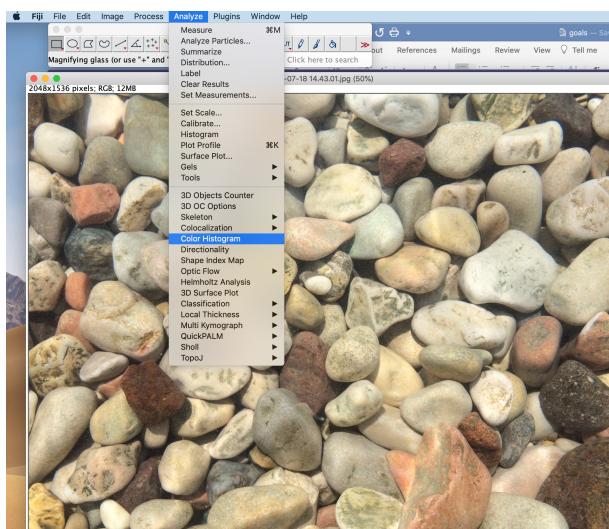
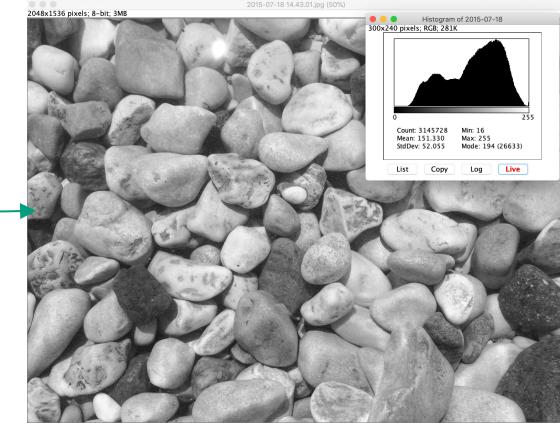
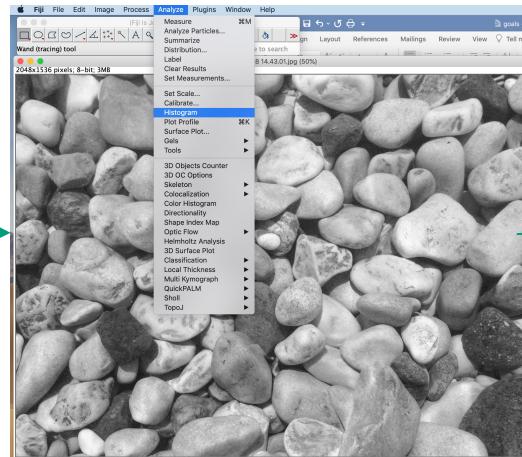
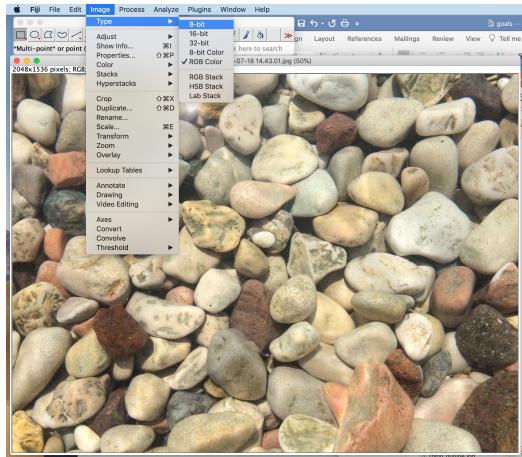
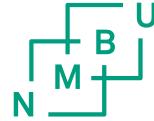
correct



(c)

Over exposed

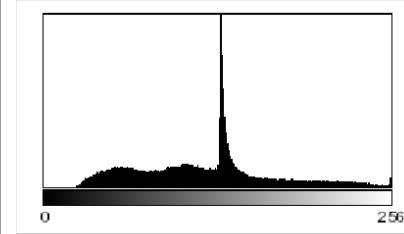
ImageJ/Fiji - Histograms



Histogram for colour image



(a)



(b) h_{Lum}



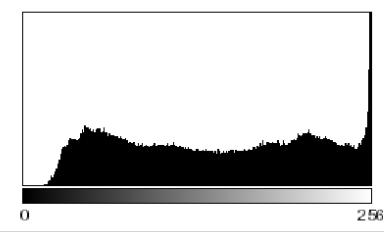
(c) R



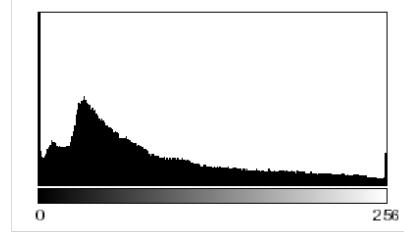
(d) G



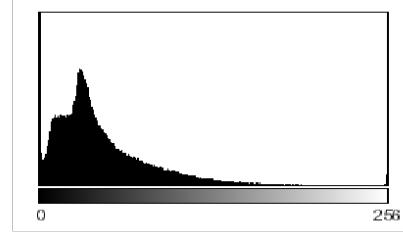
(e) B



(f) h_R

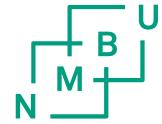


(g) h_G



(h) h_B

Python: Histogram 8-bit image



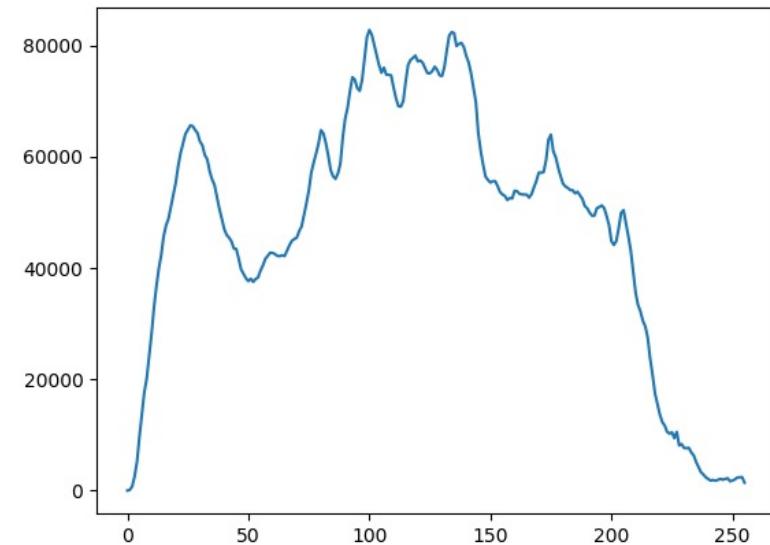
```
import matplotlib.pyplot as plt
import numpy as np

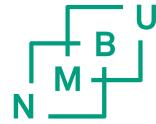
#reading image
filename = 'fall.tif'
from skimage import io
fall = io.imread(filename)

# display image
plt.imshow(fall)

#mean av 3 RGB bilder
imagemean = fall.mean(axis=2)

# histogram
shape = np.shape(imagemean)
K = 256
M = shape[0]*shape[1]
#M = shape[0]
histogram = np.zeros(256)
for i in range(shape[0]):
    for j in range(shape[1]):
        pixval = int(imagemean[i,j])
        histogram[pixval] += 1
plt.plot(histogram)
```



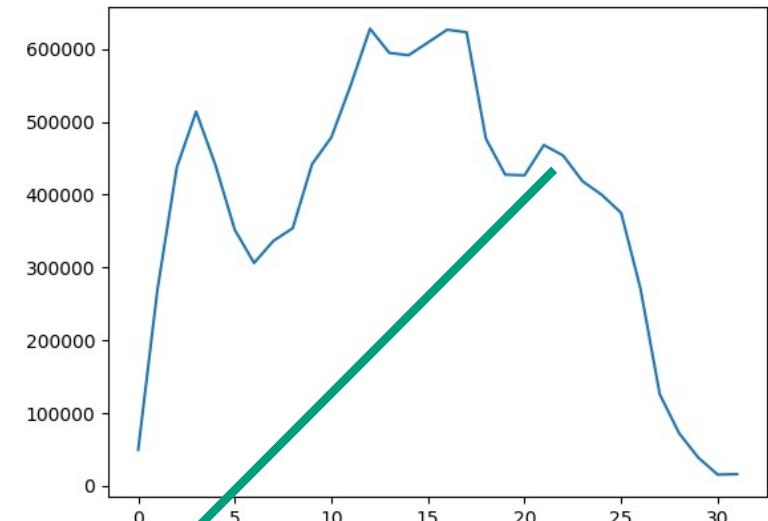
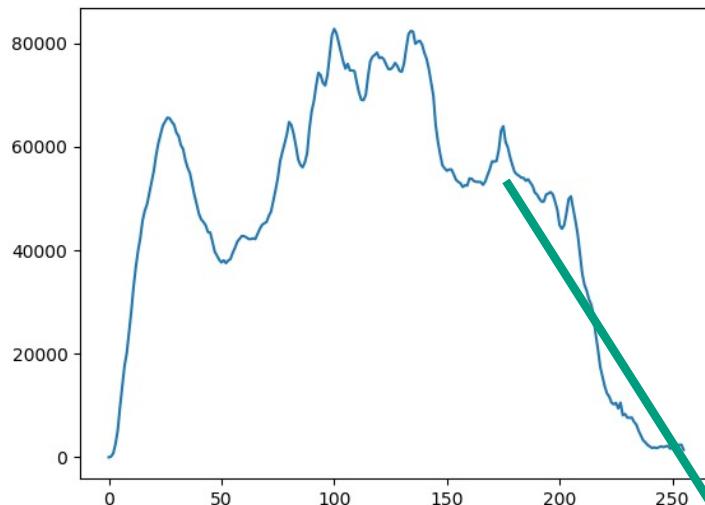


Histogram-binning, Ex 32 bins

```
# Binned histogram
K = 256
b = 32
histogram = np.zeros(b)
shape = np.shape(imagemean)
for i in range(shape[0]):
    for j in range(shape[1]):
        pixval = int(imagemean[i,j])
        histval = int(pixval*(b/K))
        histogram[histval] += 1

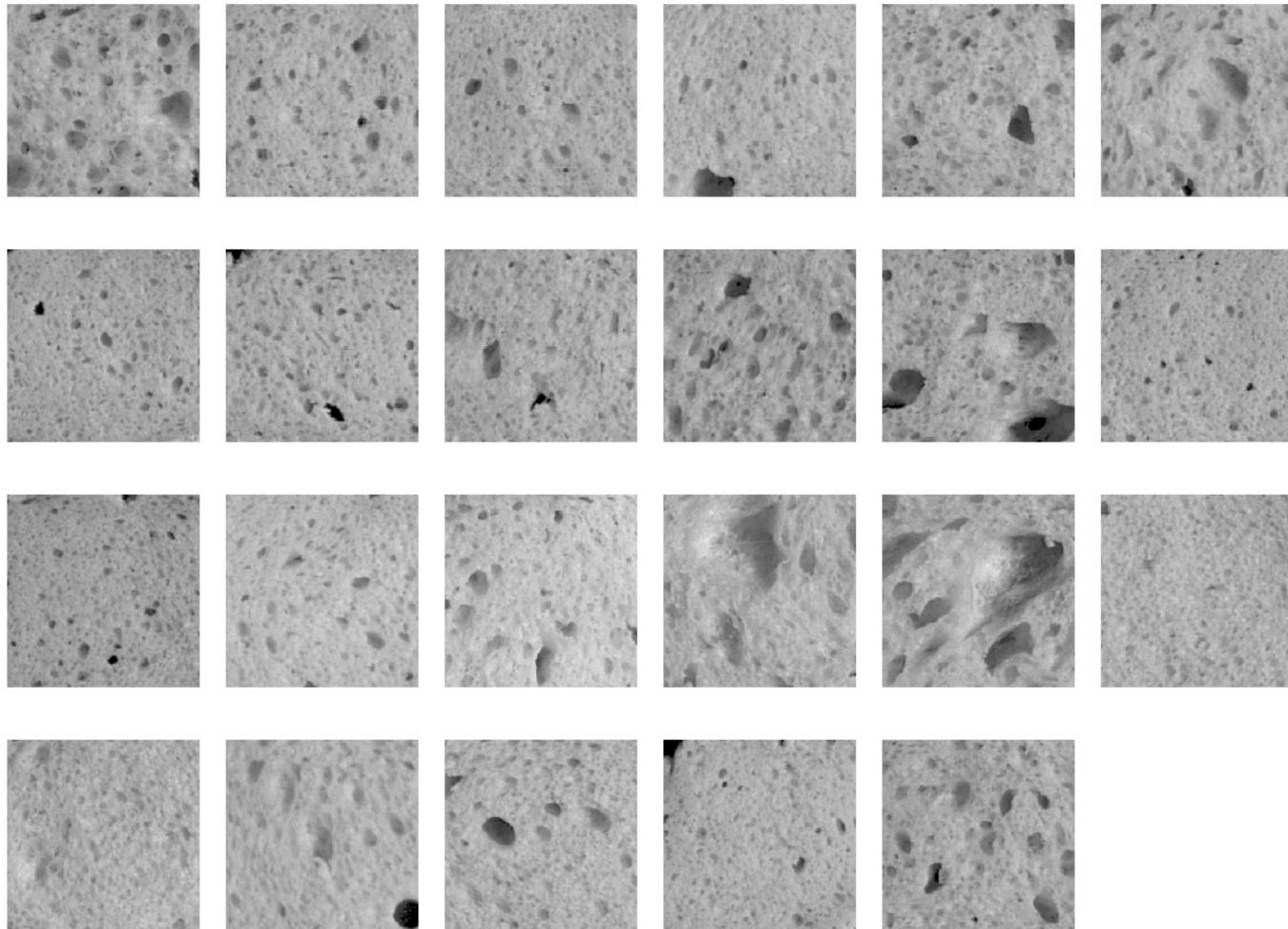
plt.figure()
plt.plot(histogram)
```

Comparison no-bin/bin



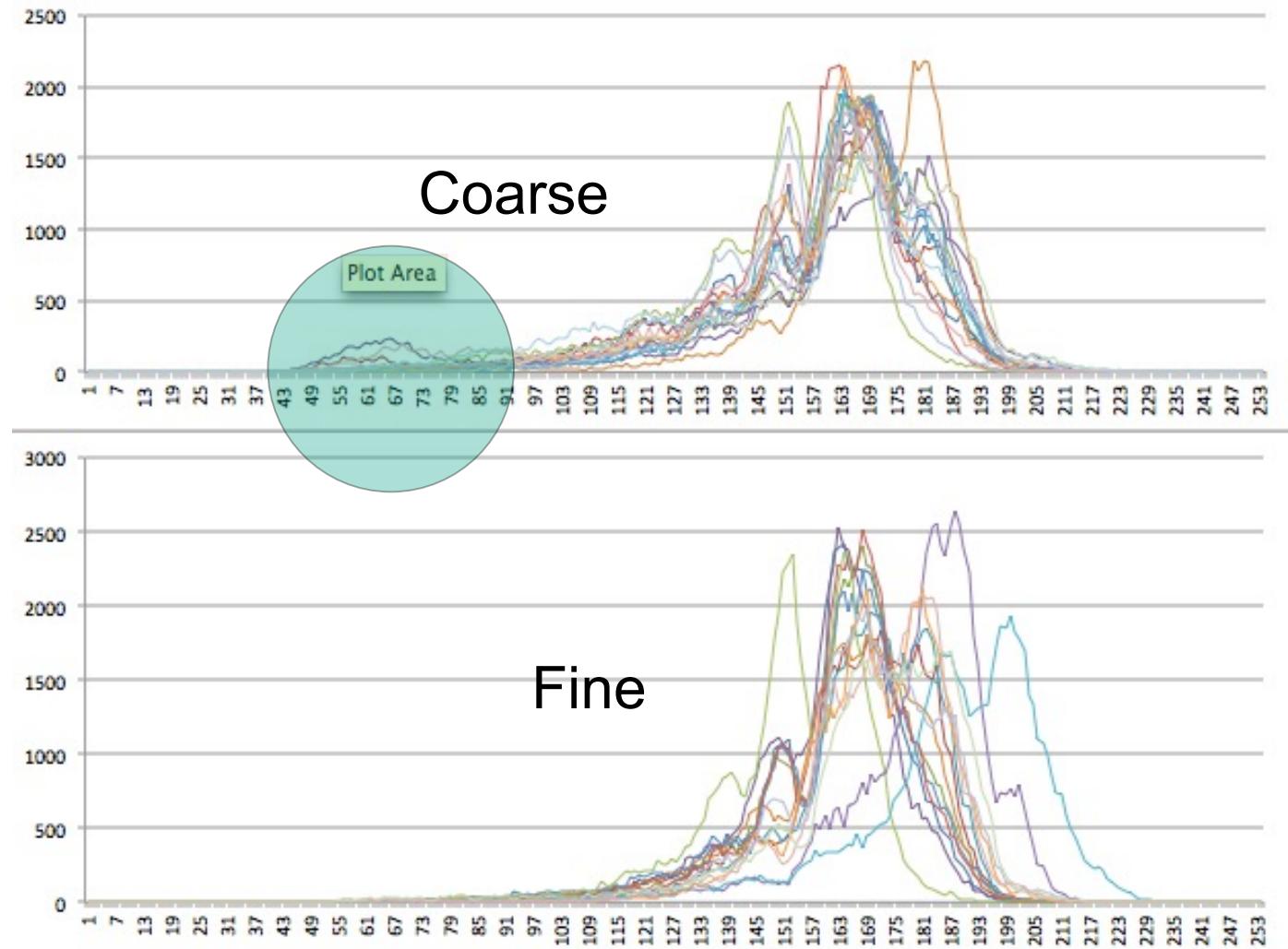
The details

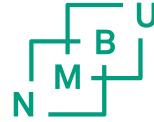
Example of histogram texture analysis



Baguette textures and histograms

N M B C



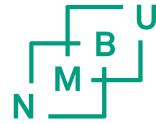


Statistics on the histogram

- Often used to describe and image
 - Max/min
 - Standard deviation
 - Variance
 - Kurtosis
 - Skewness

<http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm>

First order statistics (functions of pixel values)



- Mean

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} x_i$$

- Variance

$$\sigma^2 = \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i - \mu)^2$$

- Coefficient of variation

$$cv = \frac{\sigma}{\mu}$$

- Skewness

$$\gamma_1 = \frac{1}{(N-1)\sigma^3} \sum_{i=0}^{N-1} (x_i - \mu)^3$$

- Kurtosis

$$\gamma_2 = \frac{1}{(N-1)\sigma^4} \sum_{i=0}^{N-1} (x_i - \mu)^4 - 3$$

Skewness / Kurtosis

Skewness

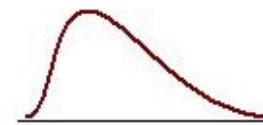
The coefficient of Skewness is a measure for the degree of symmetry in the variable distribution.



Negatively skewed distribution
or Skewed to the left
Skewness < 0



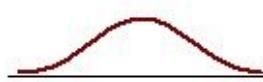
Normal distribution
Symmetrical
Skewness = 0



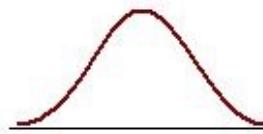
Positively skewed distribution
or Skewed to the right
Skewness > 0

Kurtosis

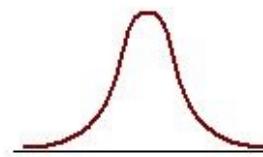
The coefficient of Kurtosis is a measure for the degree of peakedness/flatness in the variable distribution.



Platykurtic distribution
Low degree of peakedness
Kurtosis < 0



Normal distribution
Mesokurtic distribution
Kurtosis = 0



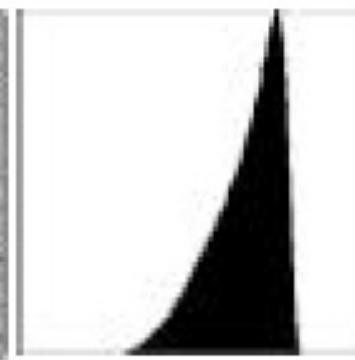
Leptokurtic distribution
High degree of peakedness
Kurtosis > 0



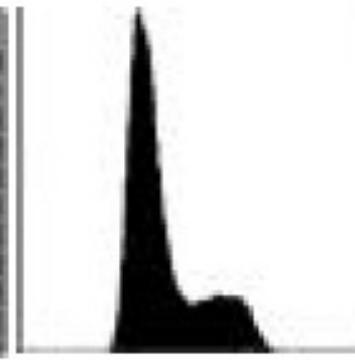
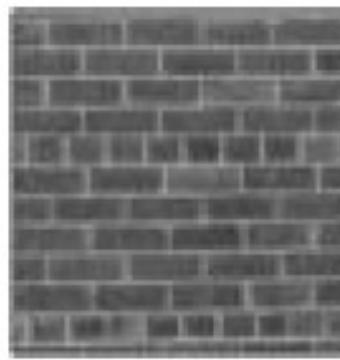
Skewness / Kurtosis

- Skewness is a measure of symmetry (or lack of symmetry)
- Kurtosis is a measure of flatness/peakness of the histogram
- Used as variables in texture analysis

Example of first order statistics



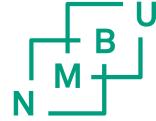
| | |
|----------|-------|
| Mean | 167.9 |
| Variance | 669.0 |
| CV | 0.15 |
| Skewness | -0.82 |
| Curtosis | 0.01 |



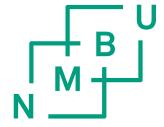
| | |
|----------|-------|
| Mean | 105.1 |
| Variance | 720 |
| CV | 0.26 |
| Skewness | 1.15 |
| Curtosis | 0.30 |

Point operations

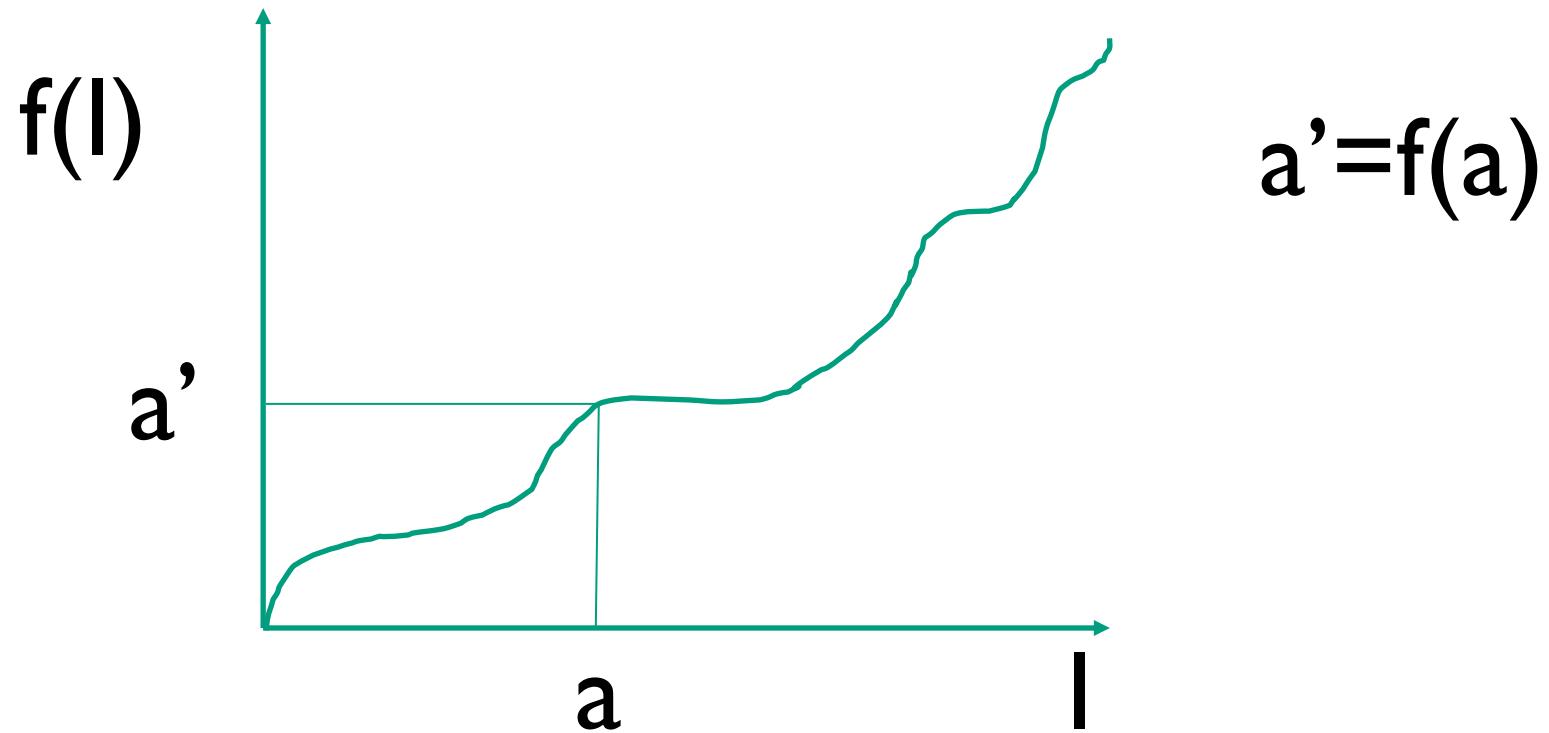
Modify with a function

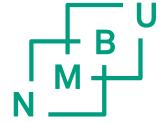


- Pixel values can be modified by a "transfer function"
 - New pixel value from a function output
 - Brightness/Contrast
 - Random transformations (curves)
 - Quantification of images (posterizing)
 - Global thresholding
 - Gamma corrections
 - Colour transformations



Transfer-function

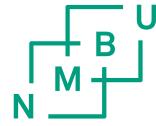




Modification of intensity

- Contrast
 - Increase contrast med 50%: $f(a) = a * 1.5$
- Brightness
 - Increase brightness with 10 units: $f(a) = a + 10$

Program: Point operation

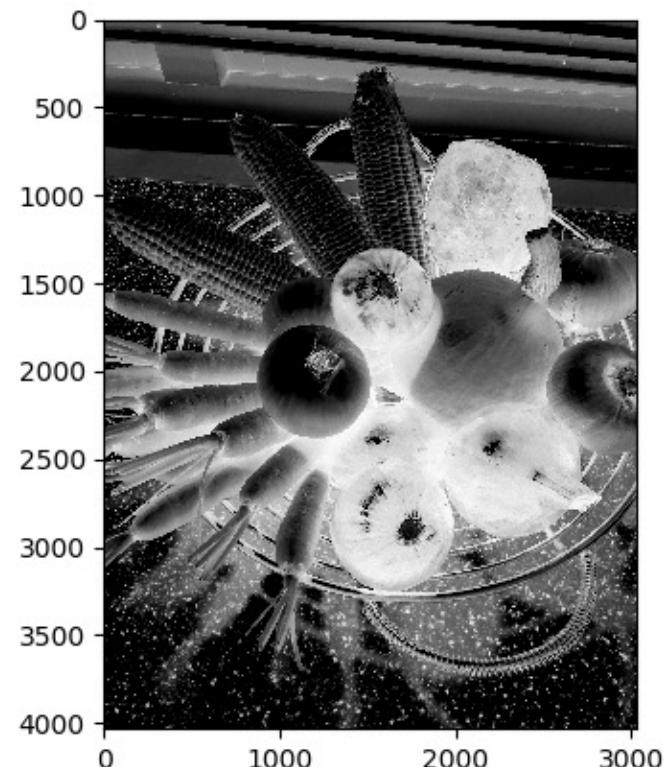
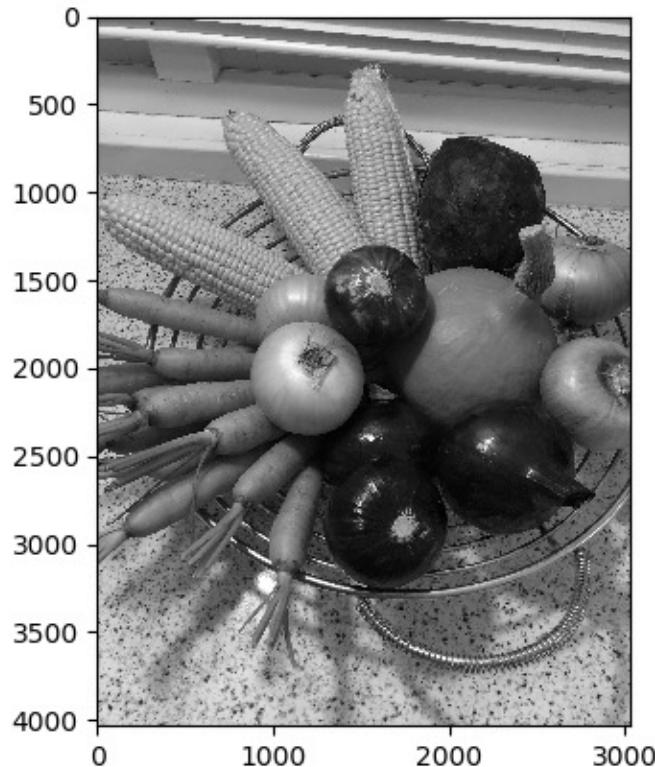
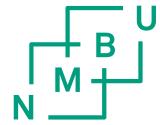


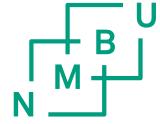
```
imagemean = lift.mean(axis=2)

shape = np.shape(imagemean)
for i in range(shape[0]):
    for j in range(shape[1]):
        imagemean[i,j] = int((imagemean[i,j])*1.5 + 0.5)
        if imagemean[i,j] > 255:
            imagemean[i,j] = 255
```

Increasing contrast with 50%, we add 0.5 to round up to an integer value

Inversion





Thresholding

$$f_{threshold}(a) = \begin{cases} a_0 & \text{for } a < a_{th} \\ a_1 & \text{for } a \geq a_{th} \end{cases}$$

- Thresholding separates the the pixel values in two classes.
- A common application is binarizing an intensity image with pixel values values 1 and 0.

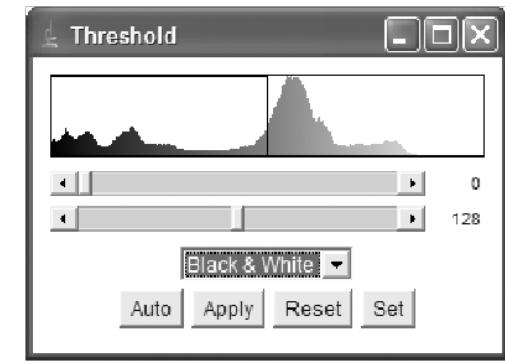
Thresholding



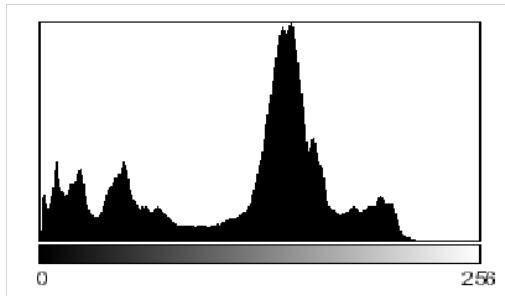
(a)



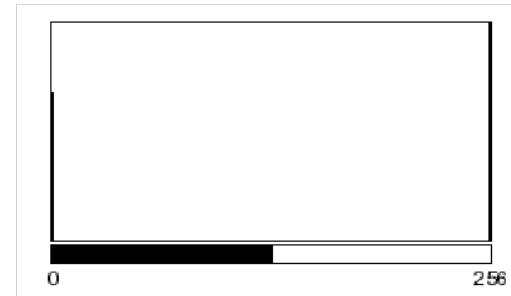
(b)



(e)

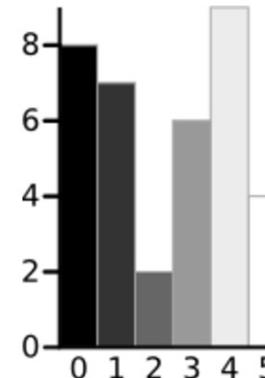


(c)



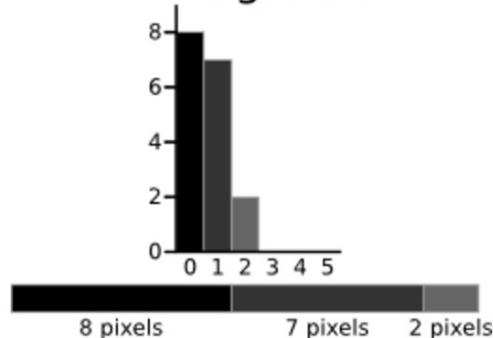
(d)

Otsu thresholding



A 6-level greyscale image and its histogram

Background

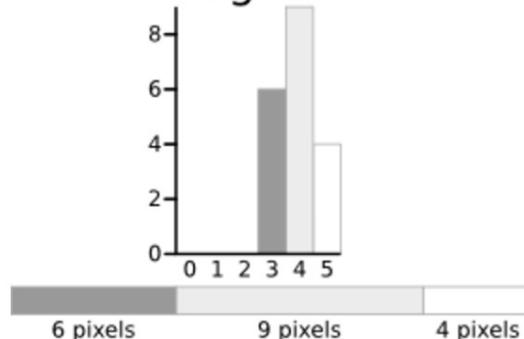


$$\text{Weight } W_b = \frac{8 + 7 + 2}{36} = 0.4722$$

$$\text{Mean } \mu_b = \frac{(0 \times 8) + (1 \times 7) + (2 \times 2)}{17} = 0.6471$$

$$\begin{aligned} \text{Variance } \sigma_b^2 &= \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17} \\ &= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17} \\ &= 0.4637 \end{aligned}$$

Foreground

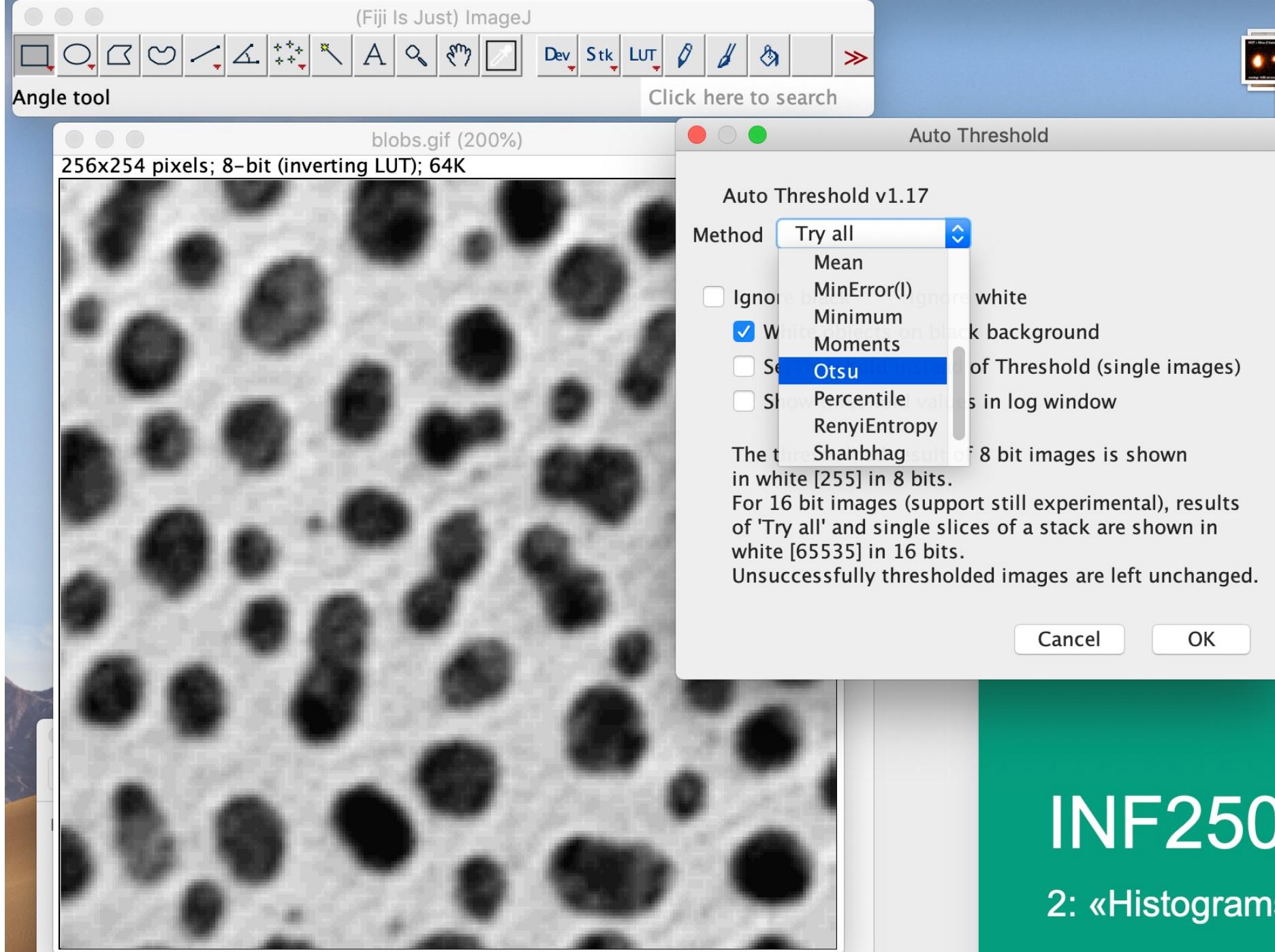


$$\text{Weight } W_f = \frac{6 + 9 + 4}{36} = 0.5278$$

$$\text{Mean } \mu_f = \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947$$

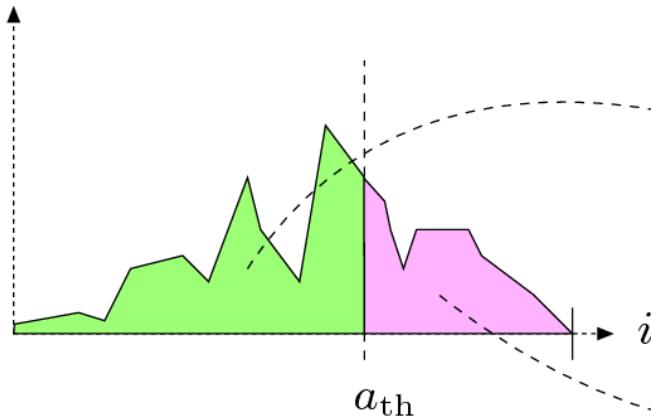
$$\begin{aligned} \text{Variance } \sigma_f^2 &= \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19} \\ &= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19} \\ &= 0.5152 \end{aligned}$$

| Threshold | T=0 | T=1 | T=2 | T=3 | T=4 | T=5 |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| Weight, Background | $W_b = 0$ | $W_b = 0.222$ | $W_b = 0.4167$ | $W_b = 0.4722$ | $W_b = 0.6389$ | $W_b = 0.8889$ |
| Mean, Background | $M_b = 0$ | $M_b = 0$ | $M_b = 0.4667$ | $M_b = 0.6471$ | $M_b = 1.2609$ | $M_b = 2.0313$ |
| Variance, Background | $\sigma^2_b = 0$ | $\sigma^2_b = 0$ | $\sigma^2_b = 0.2489$ | $\sigma^2_b = 0.4637$ | $\sigma^2_b = 1.4102$ | $\sigma^2_b = 2.5303$ |
| Weight, Foreground | $W_f = 1$ | $W_f = 0.7778$ | $W_f = 0.5833$ | $W_f = 0.5278$ | $W_f = 0.3611$ | $W_f = 0.1111$ |
| Mean, Foreground | $M_f = 2.3611$ | $M_f = 3.0357$ | $M_f = 3.7143$ | $M_f = 3.8947$ | $M_f = 4.3077$ | $M_f = 5.000$ |
| Variance, Foreground | $\sigma^2_f = 3.1196$ | $\sigma^2_f = 1.9639$ | $\sigma^2_f = 0.7755$ | $\sigma^2_f = 0.5152$ | $\sigma^2_f = 0.2130$ | $\sigma^2_f = 0$ |
| Within Class Variance | $\sigma^2_w = 3.1196$ | $\sigma^2_w = 1.5268$ | $\sigma^2_w = 0.5561$ | $\sigma^2_w = 0.4909$ | $\sigma^2_w = 0.9779$ | $\sigma^2_w = 2.2491$ |

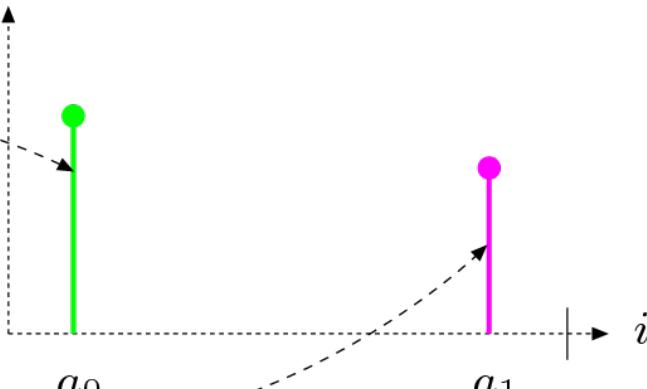


Histogram after thresholding

$h(i)$



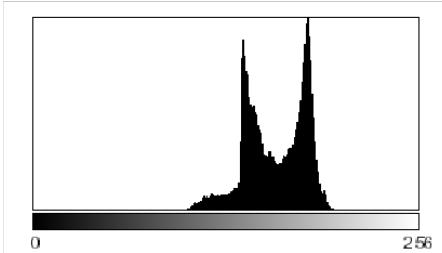
$h'(i)$



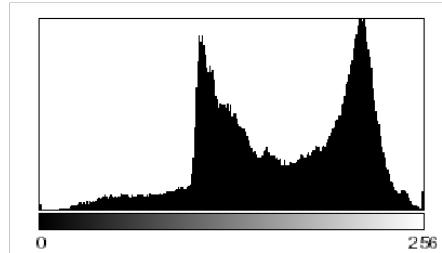
(a)

(b)

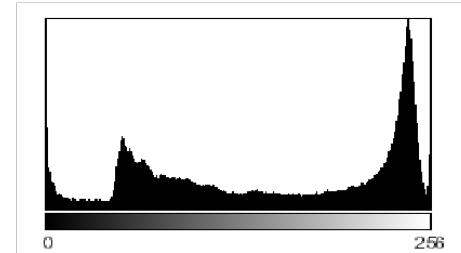
Contrast



(a)

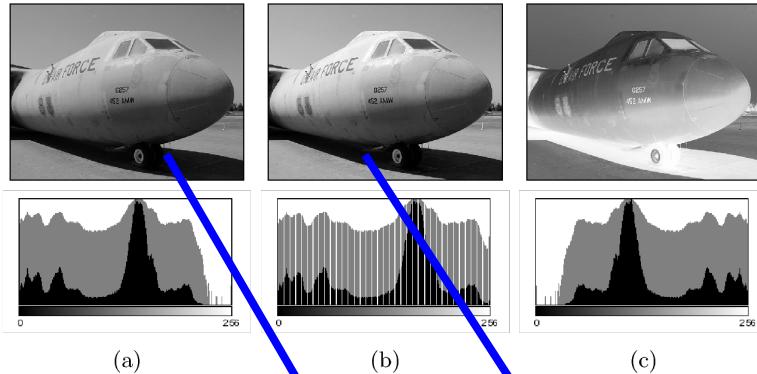


(b)



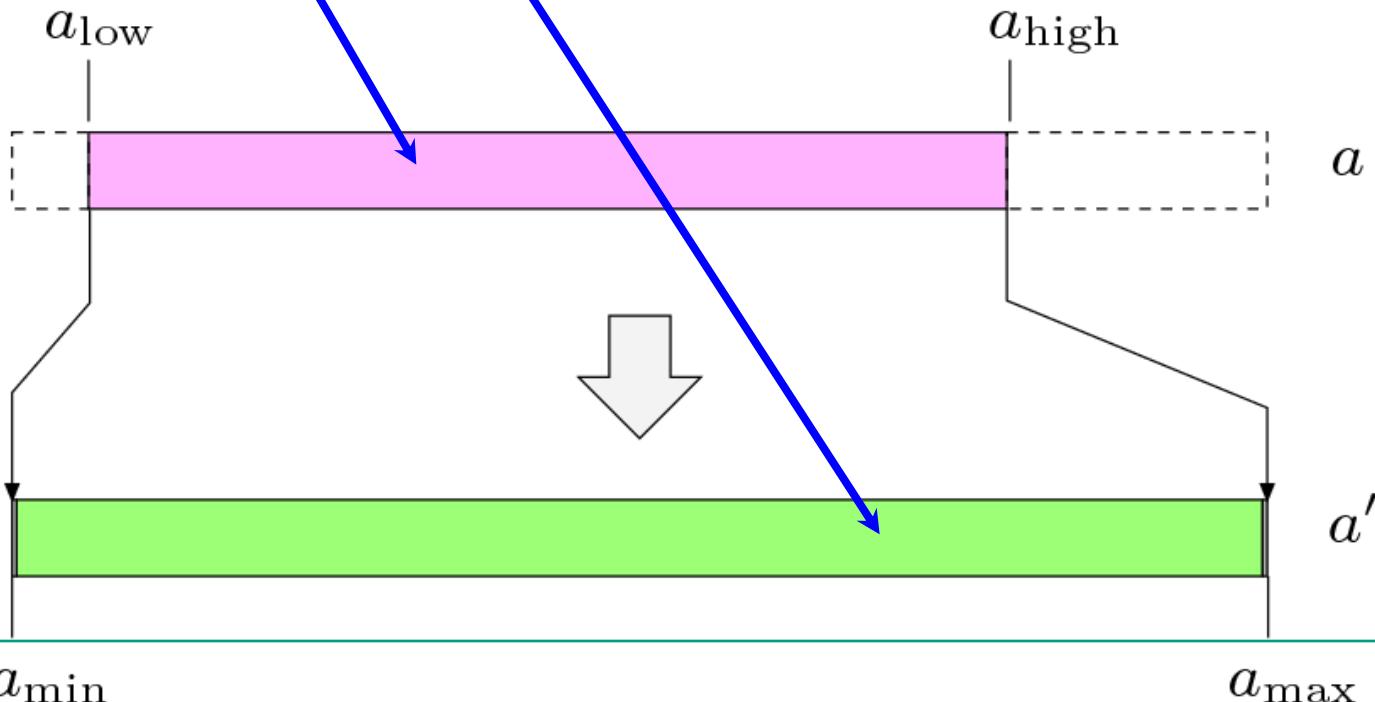
(c)

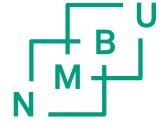
Automatic contrast adjustment



$$f_{ac}(a) = a_{\min} + (a - a_{\min}) \cdot \frac{a_{\max} - a_{\min}}{a_{\text{high}} - a_{\min}}$$

$$f_{ac}(a) = (a - a_{\min}) \cdot \frac{255}{a_{\text{high}} - a_{\min}}$$



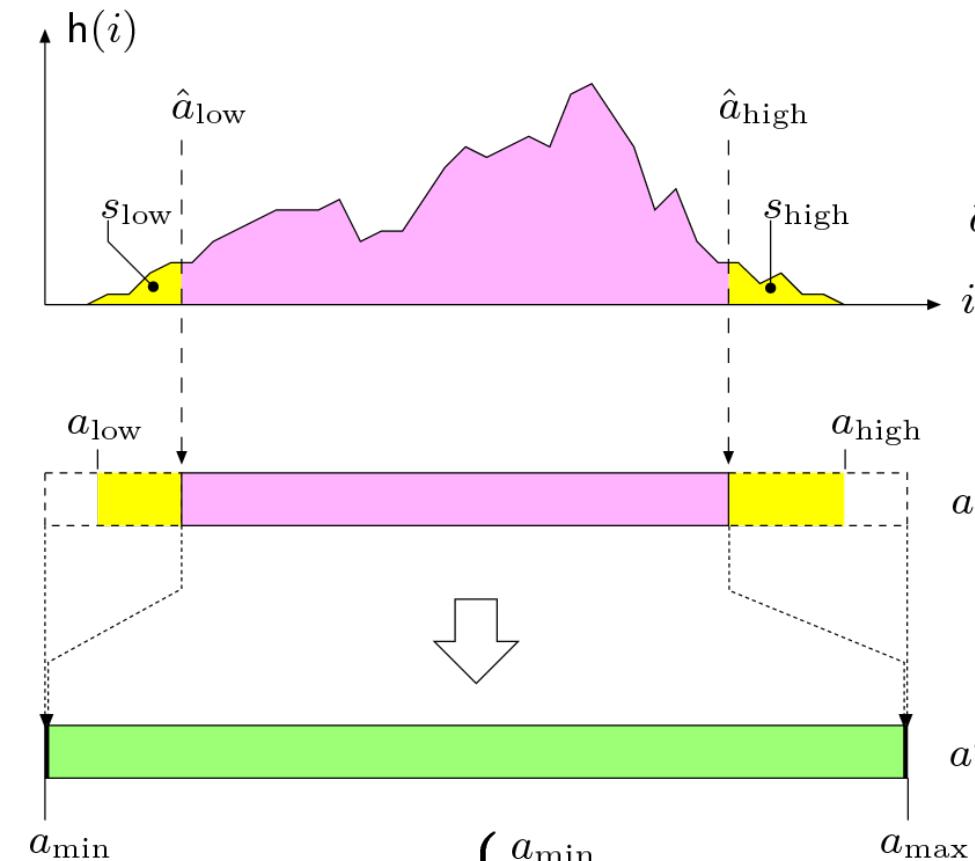
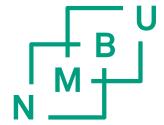


Automatic contrast adjustment

```
shape = np.shape(ireland)
ima = ireland
amin = 0
amax = 255
alow = int(min(ima.flatten()))
ahigh = int(max(ima.flatten()))
for i in range(shape[0]):
    for j in range(shape[1]):
        ima[i,j] = amin+(ima[i,j]-alow)*(amax-amin)/(ahigh-alow)

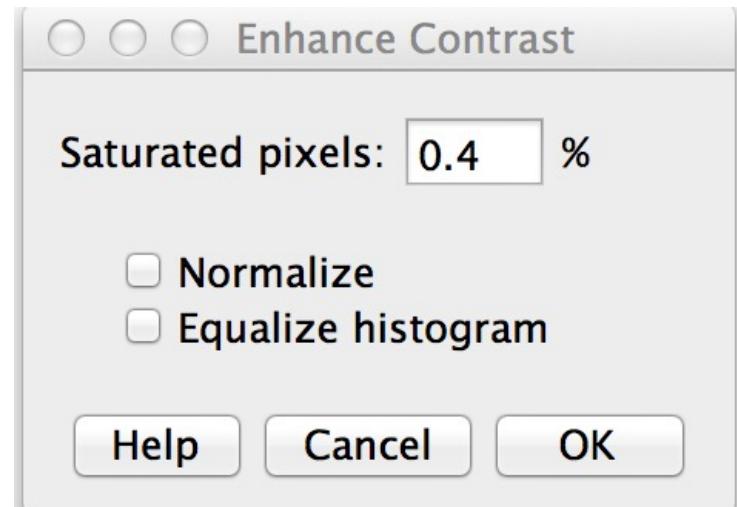
plt.imshow(ima, 'gray')
plt.hist(ima.ravel(),256,[0,256])
plt.show()
```

Modified automatic contrast adjustment



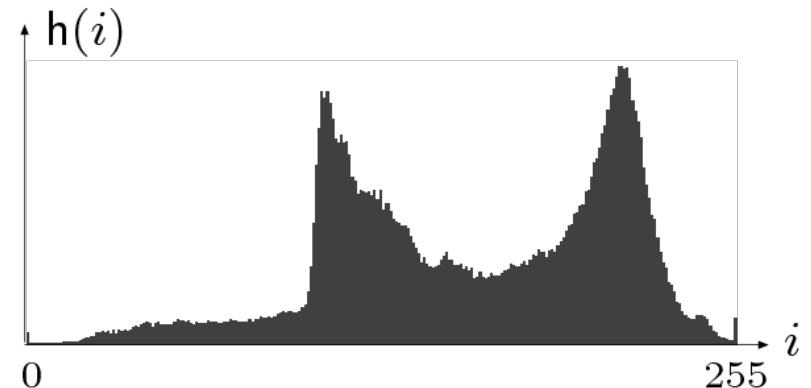
$$\hat{a}_{\text{low}} = \min \{ i \mid H(i) \geq M \cdot N \cdot s_{\text{low}} \}$$

$$\hat{a}_{\text{high}} = \max \{ i \mid H(i) \leq M \cdot N \cdot (1 - s_{\text{high}}) \}$$



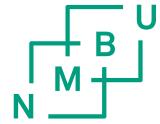
$$f_{\text{mac}}(a) = \begin{cases} a_{\text{min}} & \text{for } a \leq \hat{a}_{\text{low}} \\ a_{\text{min}} + (a - \hat{a}_{\text{low}}) \cdot \frac{a_{\text{max}} - a_{\text{min}}}{\hat{a}_{\text{high}} - \hat{a}_{\text{low}}} & \text{for } \hat{a}_{\text{low}} < a < \hat{a}_{\text{high}} \\ a_{\text{max}} & \text{for } a \geq \hat{a}_{\text{high}} \end{cases}$$

Cumulative histogram



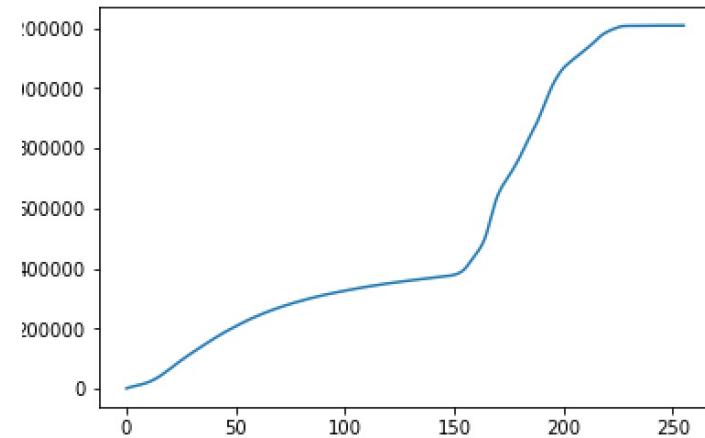
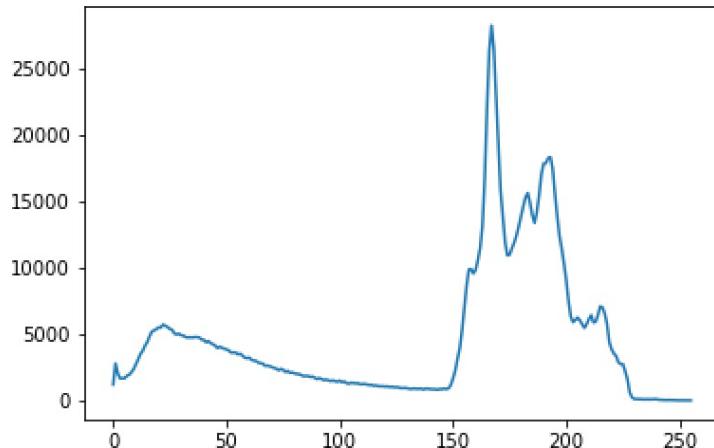
$$H(i) = \sum_{j=0}^i h(j) \quad \text{for } 0 \leq i < K$$

Cumulative histogram

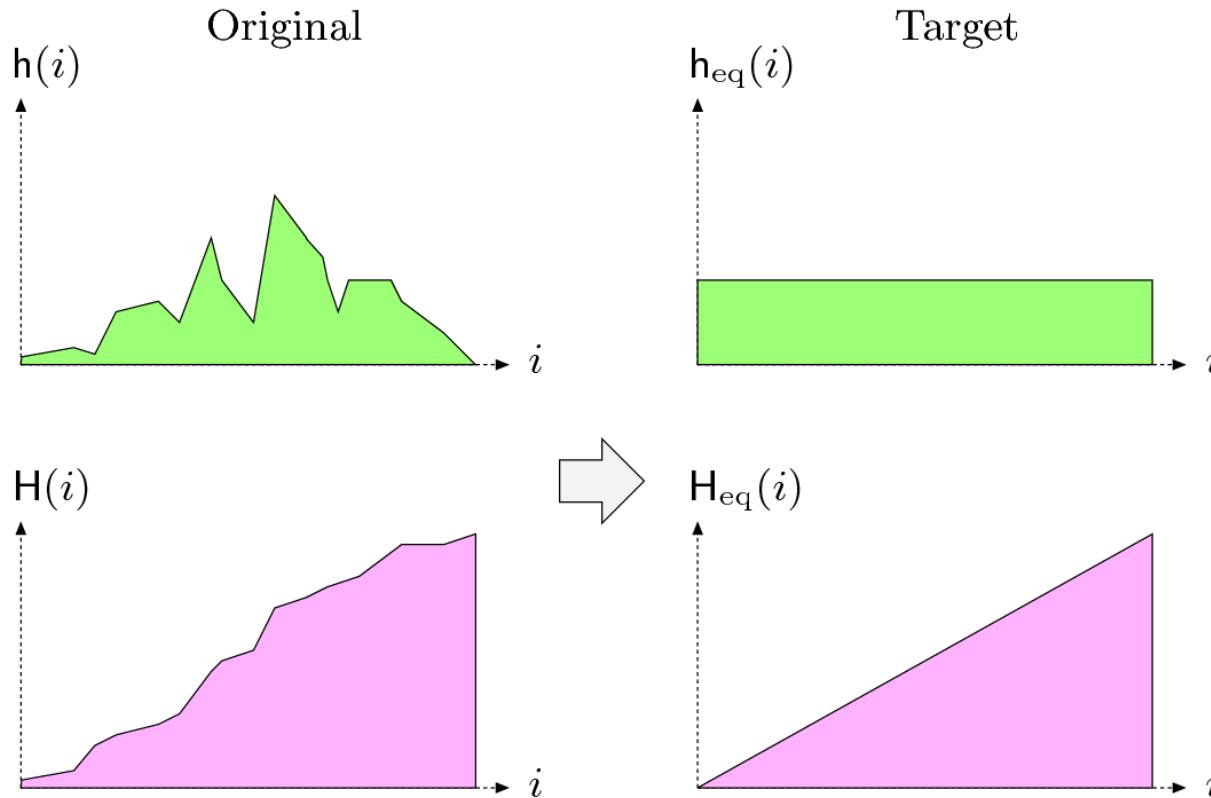


```
# cumulative hist
cumhist = np.zeros(256)
cumhist[0] = histogram[0]
for i in range(255):
    cumhist[i+1] = cumhist[i]+histogram[i+1]

plt.plot(cumhist)
```



Histogram equalisation



Histogram equalisation

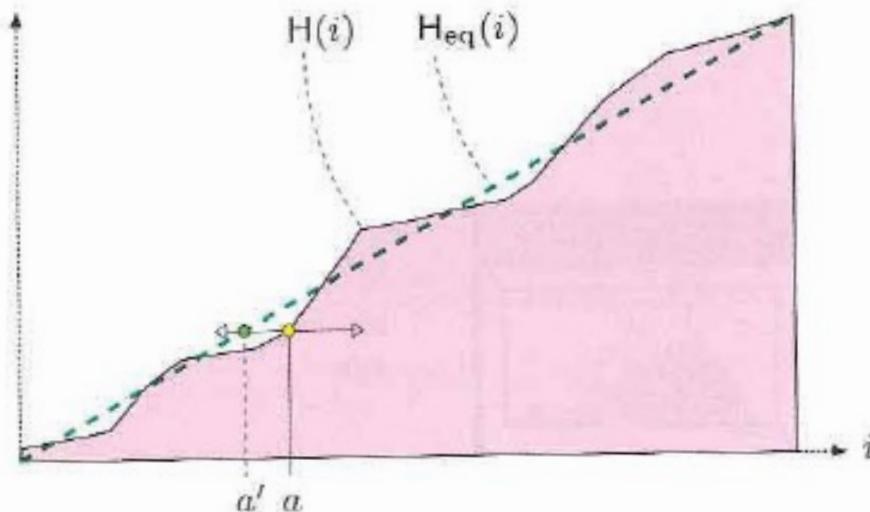


Figure 4.9 Histogram equalization on the cumulative histogram. A suitable point operation $a' \leftarrow f_{\text{eq}}(a)$ shifts each histogram line from its original position a to a' (left or right) such that the resulting cumulative histogram H_{eq} is approximately linear.

$$f_{\text{eq}}(a) = \left\lfloor H(a) \cdot \frac{K-1}{MN} \right\rfloor \quad \begin{array}{l} \text{Range}=[0, K-1] \\ \text{M}= \text{height} \\ \text{N}= \text{width} \end{array}$$

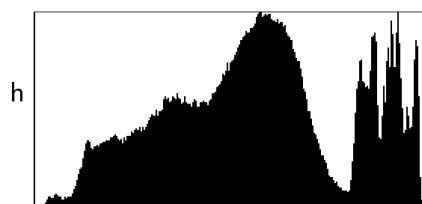
Histogram equalisation



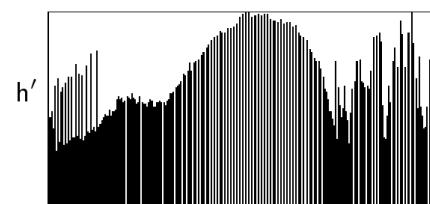
(a)



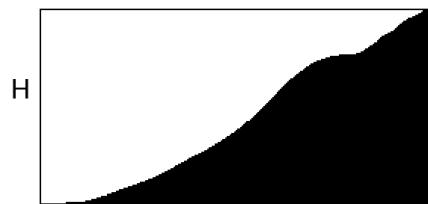
(b)



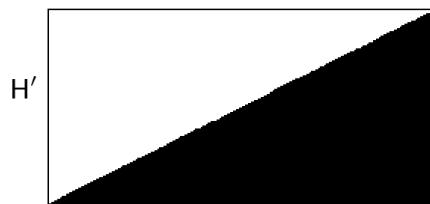
(c)



(d)



(e)

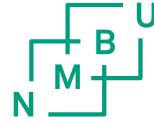


(f)

$$\tilde{H}(i) = \sum_{j=0}^i \sqrt{H(j)}$$

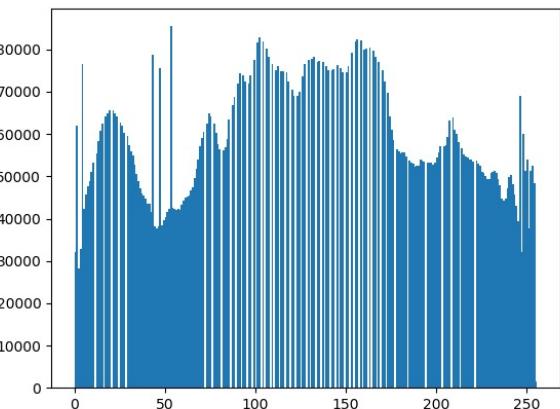
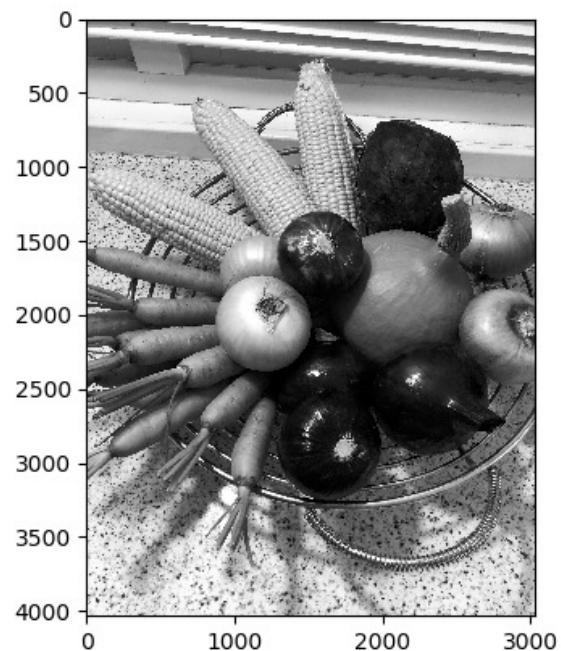
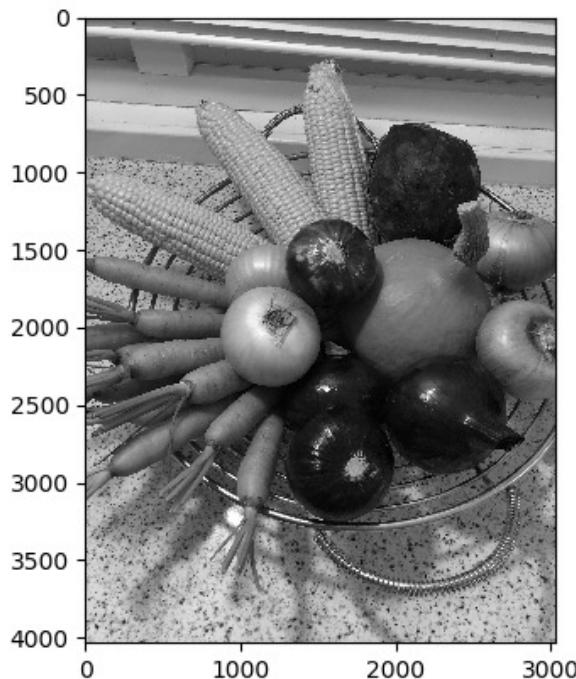
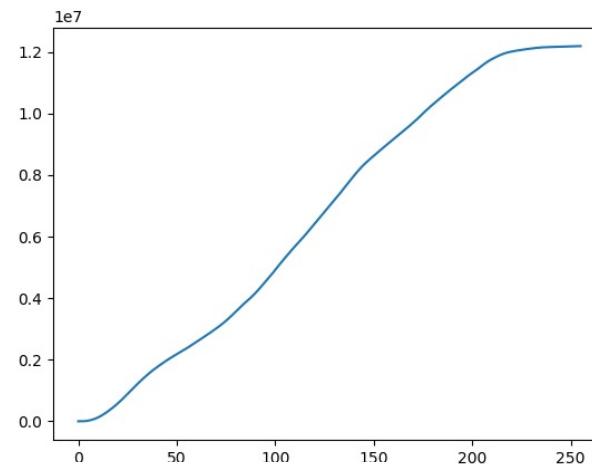
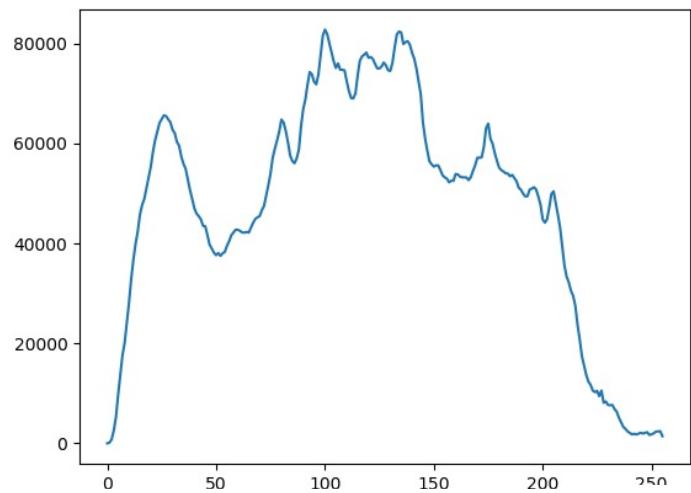
$$f_{\text{eq}}(a) = \left\lfloor H(a) \cdot \frac{K-1}{MN} \right\rfloor$$

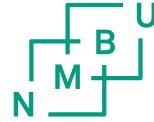
Program: Histogram equalisation



```
# cumulative hist
cumhist = np.zeros(256)
cumhist[0] = histogram[0]
for i in range(255):
    cumhist[i+1] = cumhist[i]+histogram[i+1]

# equalisation
for i in range(shape[0]):
    for j in range(shape[1]):
        a = int(ima[i,j])
        b = cumhist[a]*(K-1)/M
        ima[i,j] = b
```





Histogram specification

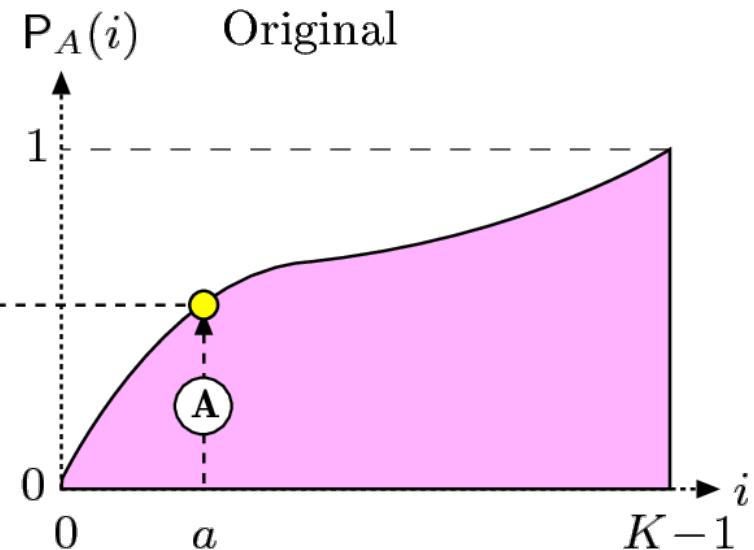
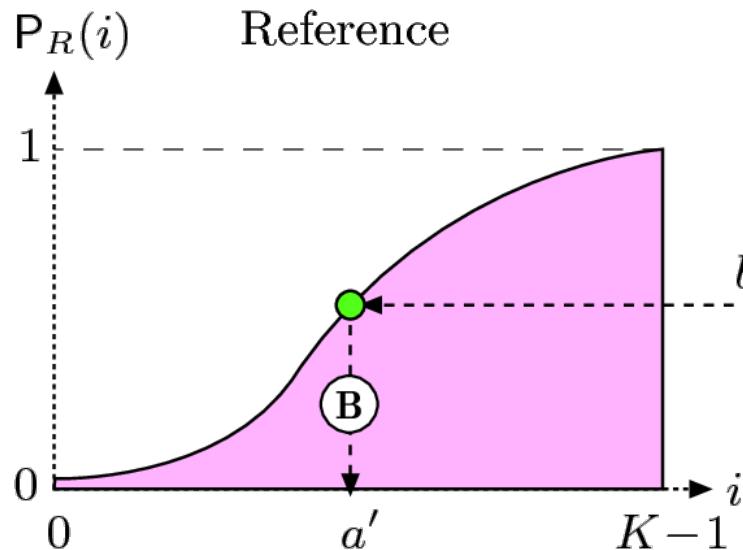
- Histogram specification can be approached through the cumulated histogram
- We have the original image A with histogram distribution P_A
- We have a reference image R with histogram distribution P_R
- We transform A so that the histogram P_A is as equal as possible to the histogram P_R

$$P_{A'}(i) \approx P_R(i) \quad \text{for } 0 \leq i < K$$

The pixel value a is transformed to a'

$$a' = P_R^{-1}(P_A(a))$$

Histogram specification



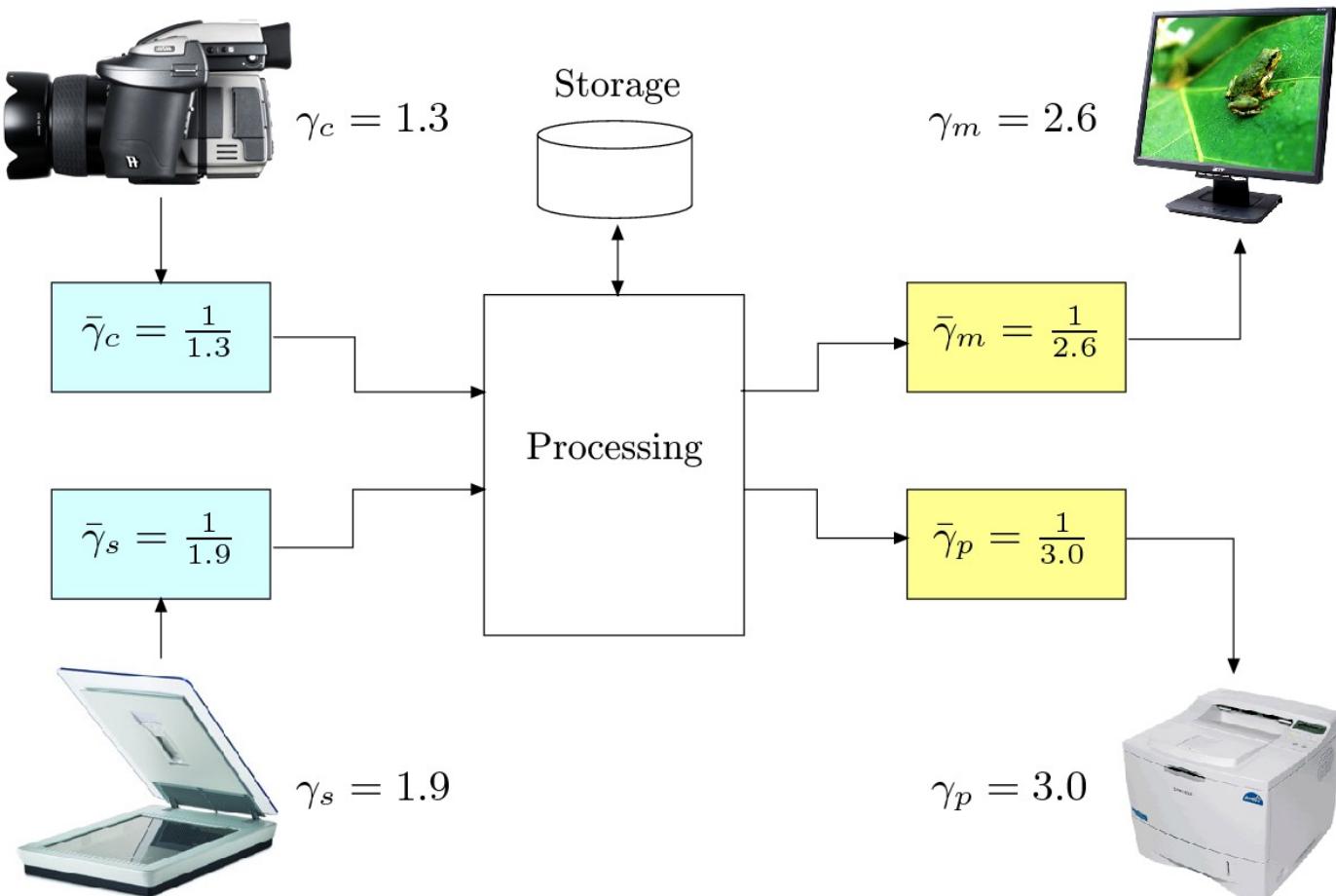
$$a' = f_{\text{hs}}(a)$$

$$f_{\text{hs}}(a) = a' = P_R^{-1}(P_A(a))$$

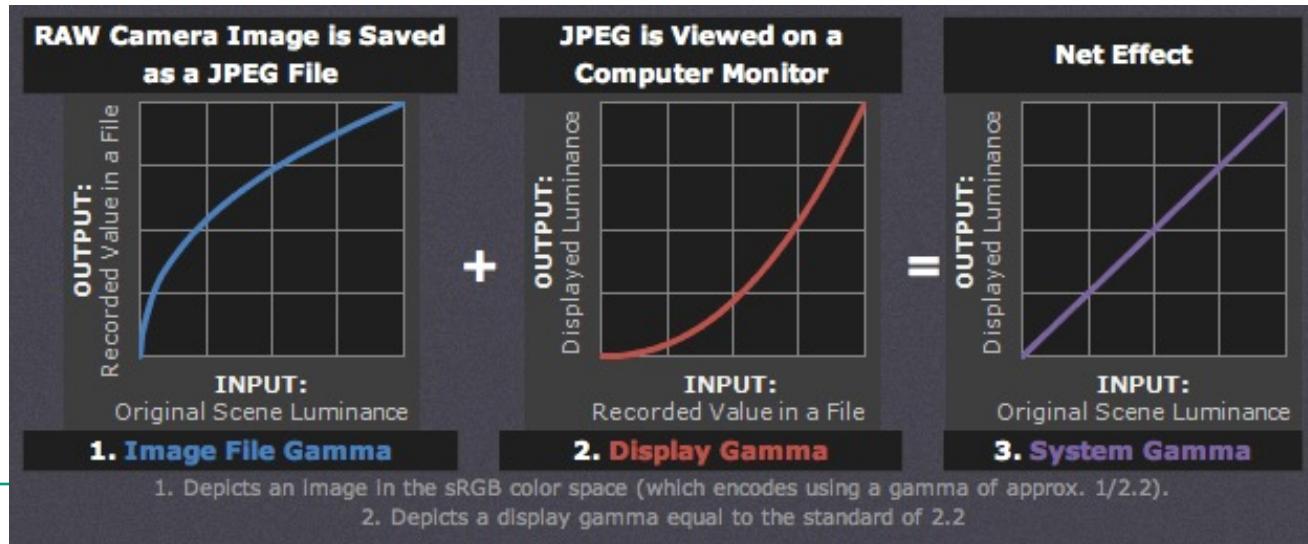
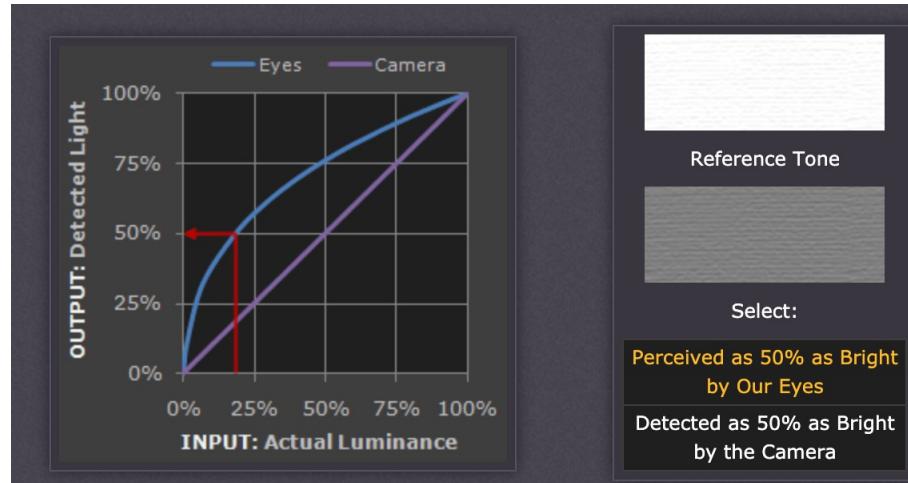
$$a' = P_R^{-1}(P_A(a))$$

$$P_{A'}(i) \approx P_R(i) \quad \text{for } 0 \leq i < K$$

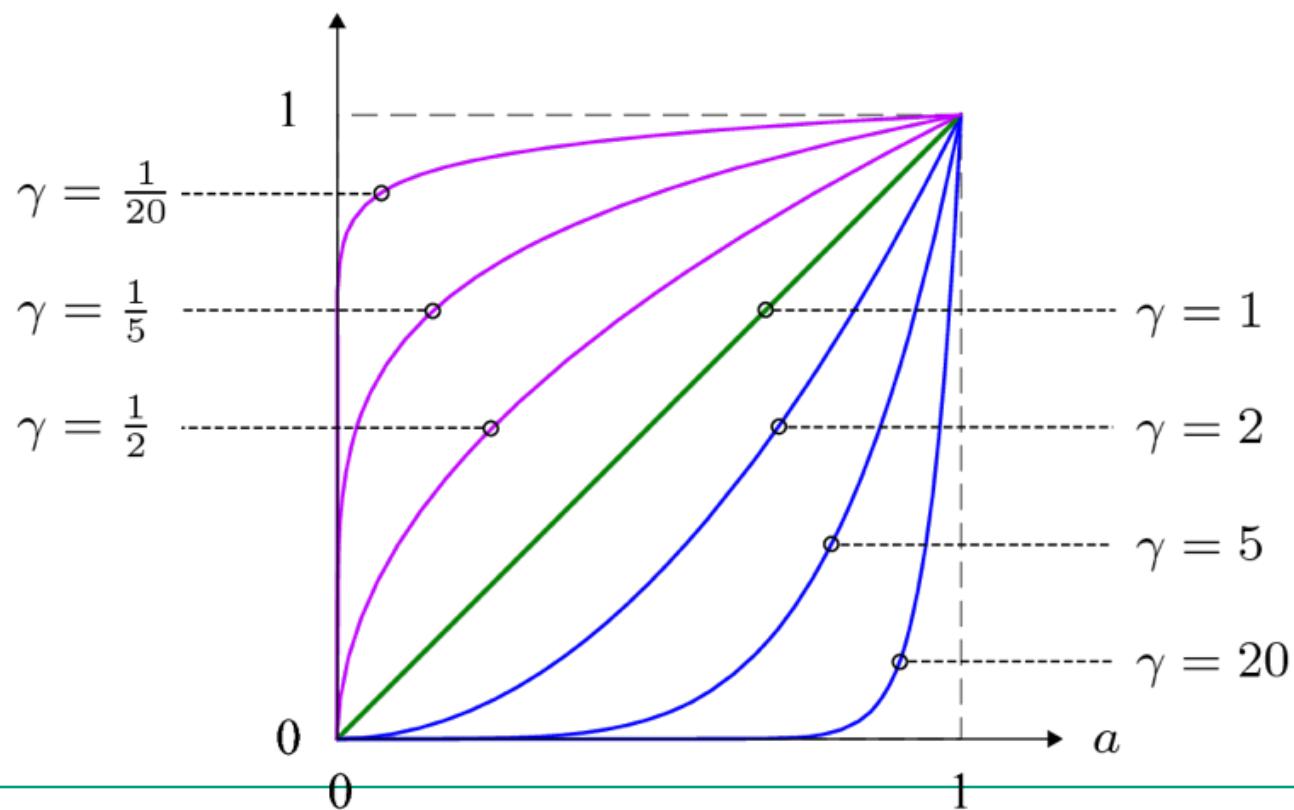
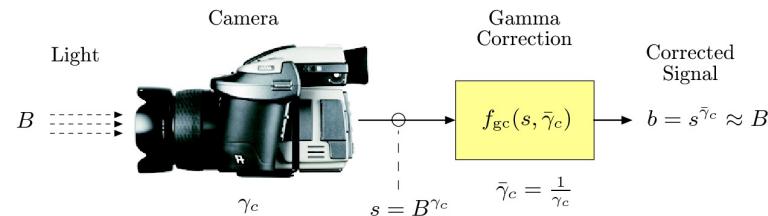
Gamma scale



Gamma scale



GAMMA



«Using gamma» in an artistic way

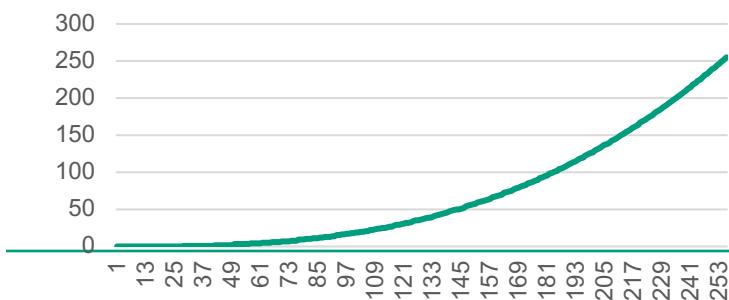


Original

Gamma = 0.2

Gamma 0.2
Histogram eq.

Difference between
Original and
Gamma 0.2
Histogram eq.



Gamma in ImageJ: Process – Math - Gamma

ALPHA BLENDING



- A technique to play with transparency between two images

$$I'(u, v) \leftarrow \alpha \cdot I_{\text{BG}}(u, v) + (1 - \alpha) \cdot I_{\text{FG}}(u, v)$$



I_{BG}



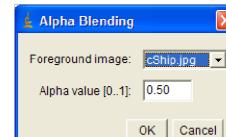
$\alpha = 0.25$



I_{FG}



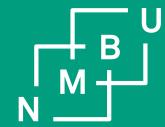
$\alpha = 0.50$



GenericDialog



$\alpha = 0.75$



INF250

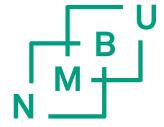
3 Filter



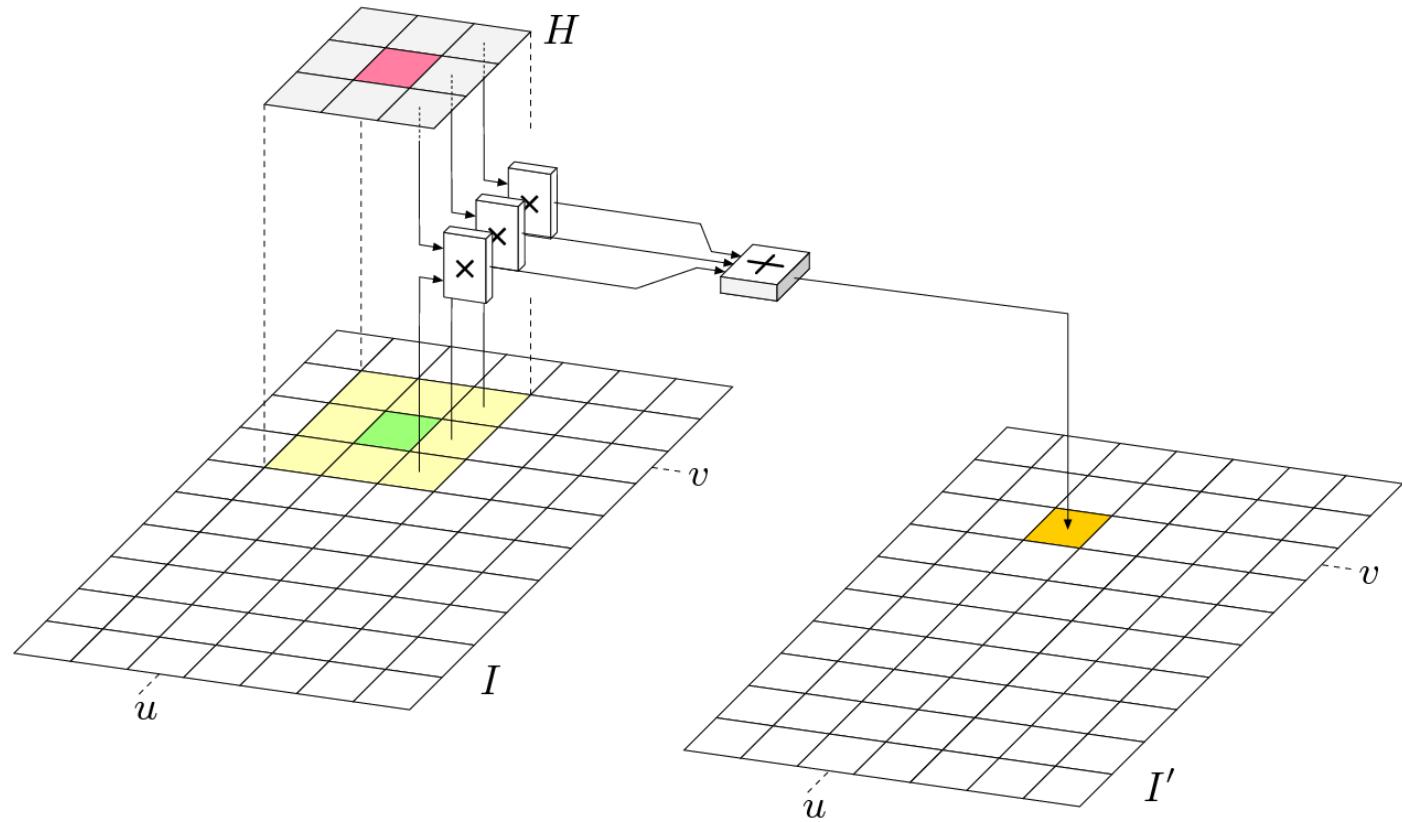
Filter or point operation

- The main difference between filter and point operation
 - Filter: Computations involve several pixels from the source
 - Point operations: Computations involve only one point (pixel) from the source
 - To blur (unsharpen) or sharpen an image you need to use a filter operation

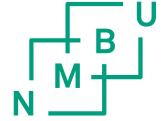




Filtering



Smoothing

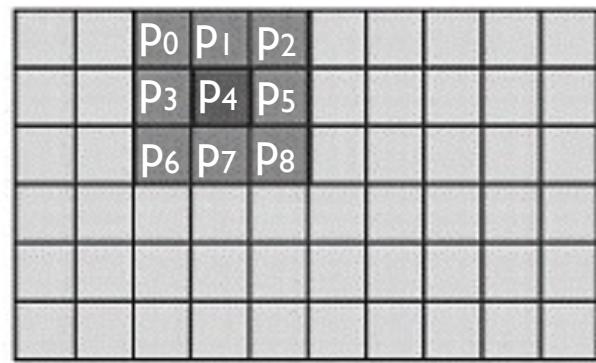
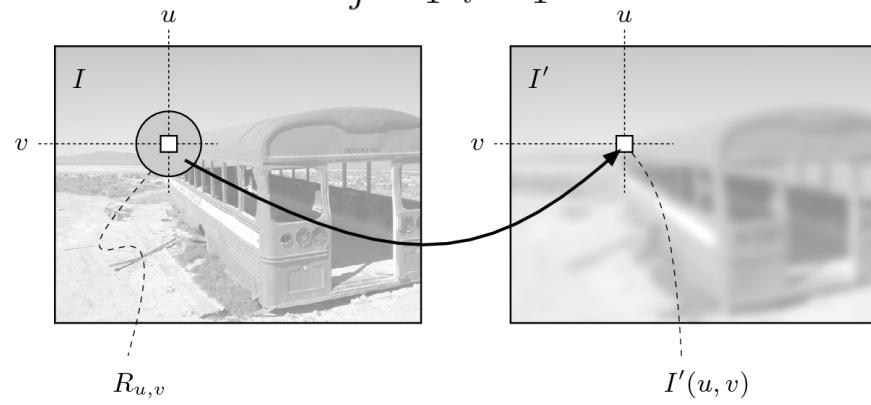


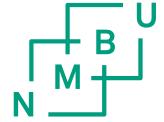
MEAN filter

$$I'(u, v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

$$I'(u, v) \leftarrow \frac{1}{9} \cdot [I(u-1, v-1) + I(u, v-1) + I(u+1, v-1) + I(u-1, v) + I(u, v) + I(u+1, v) + I(u-1, v+1) + I(u, v+1) + I(u+1, v+1)]$$

$$I'(u, v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^1 \sum_{i=-1}^1 I(u+i, v+j)$$





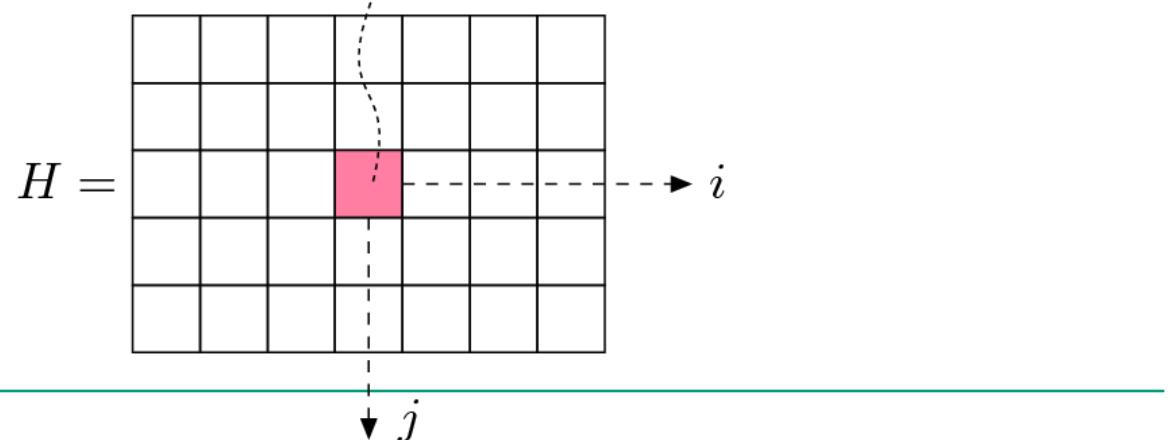
Smoothing filter

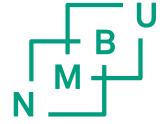
$$\begin{array}{c}
 H(i, j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \\
 \text{filter operator}
 \end{array}$$

$$I'(u, v) \leftarrow \sum_{(i, j) \in R_H} I(u + i, v + j) \cdot H(i, j)$$

$(0, 0) = \text{Hot Spot}$

Convolution between
filter and pixel value



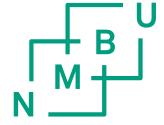


Linear filters

Linear filters combine pixel values in a linear way, i.e., as a weighted summation. The results from a linear filter is completely specified by the coefficients of the filter matrix.

The following steps are performed:

1. The filter matrix H is moved over the original image I such that its origin $H(0,0)$ coincides with the current image position (u,v)
2. All filter coefficients $H(i,j)$ are multiplied with the corresponding image element $I(u+i, v+j)$ and the results are added
3. The resulting sum is stored at the current position in the new image $I'(u,v)$



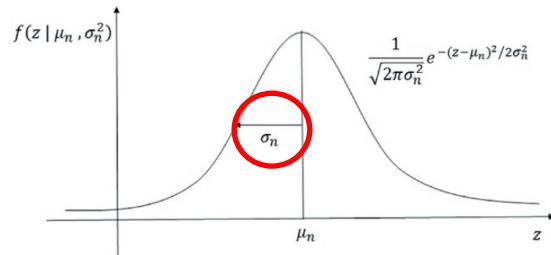
Linear convolution

$$I'(u, v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u + i, v + j) \cdot H(i, j)$$

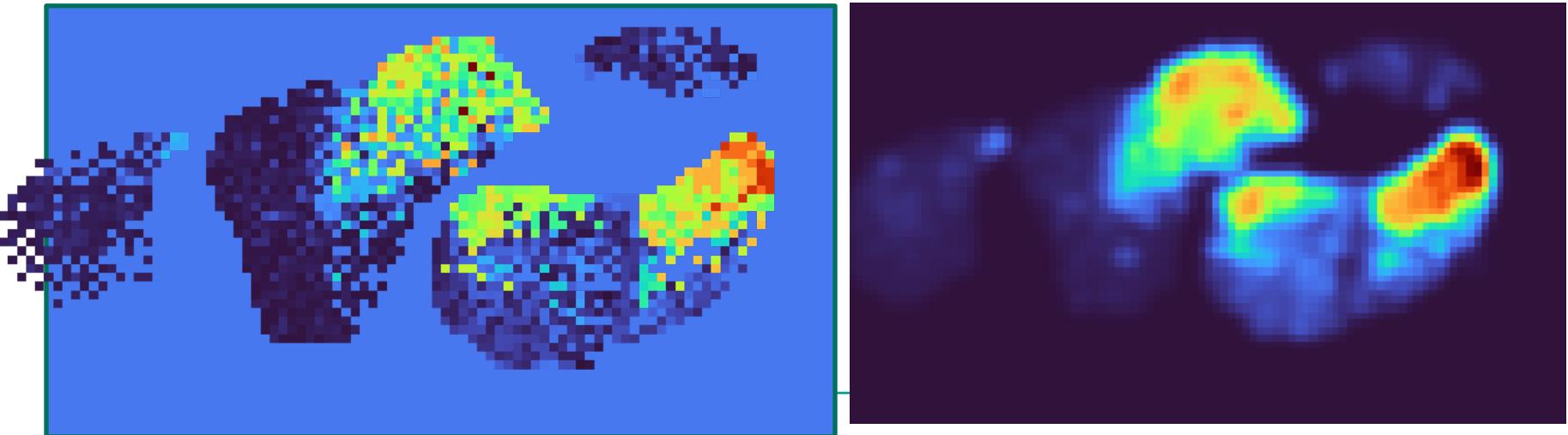
Clown example ...

Convolution

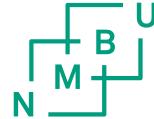
Gaussian blur



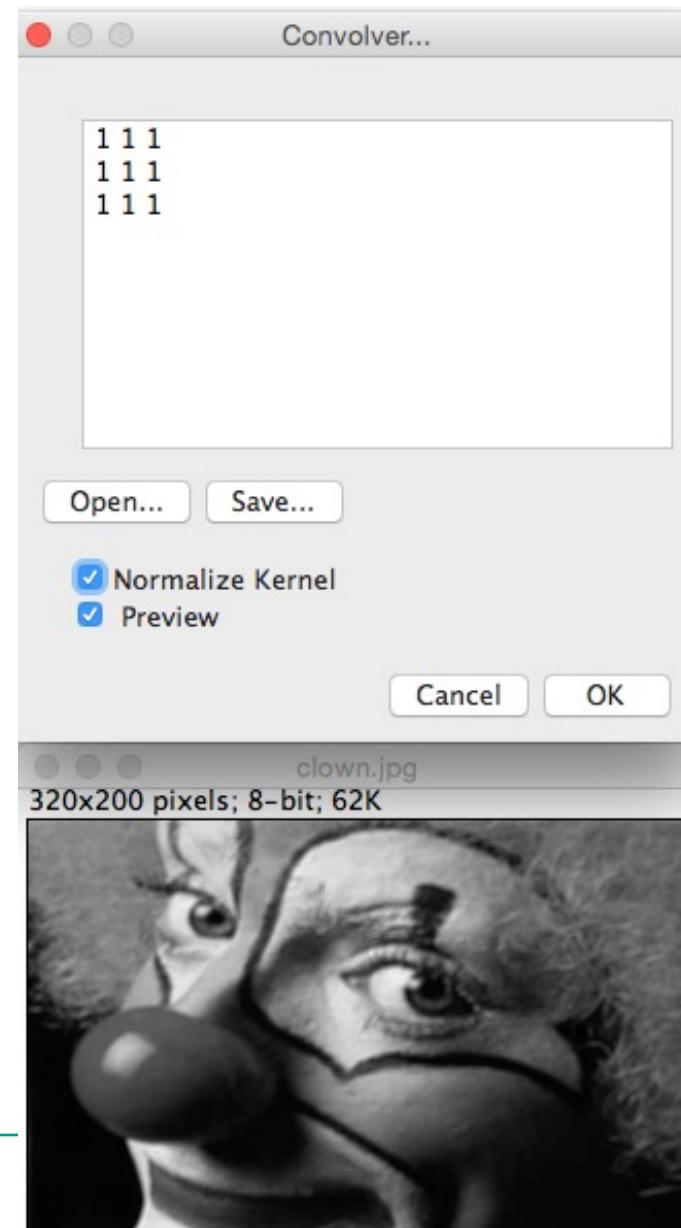
...

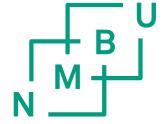


Smoothing in ImageJ



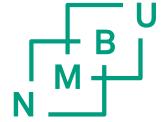
- Process/Filters/Convolve
- Edit filter
- Use Normalize Kernel
- Preview shows the result
- Save the filter





Properties for linear convolution

- Commutativity
 - Linearity
 - Associativity
 - x/y separability
-



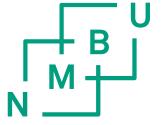
Commutativity

Linear convolution is commutative, i.e.,

$$I^*H = H^*I$$

$$\begin{array}{ccc} \begin{array}{c} \text{orange dots} \\ + \\ \text{blue dots} \end{array} & \begin{array}{c} \text{blue dots} \\ + \\ \text{orange dots} \end{array} \\ 6 + 3 = 9 = 3 + 6 \\ \begin{array}{c} 4 \times 2 \\ = \\ 8 \end{array} & \begin{array}{c} 2 \times 4 \\ = \\ 8 \end{array} \end{array}$$

Linearity

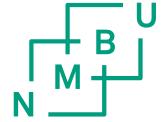


$$(s \cdot I) * H = s \cdot (I * H)$$

$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$

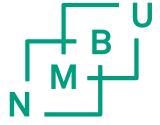
$$(b + I) * H \neq b + (I * H)$$

Assosiativity



The order of the successive filter operations is irrelevant

$$A * (B * C) = (A * B) * C$$



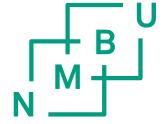
Separability

If $H = H_1 * H_2 * \dots * H_n$

then

$$I * H = I * (H_1 * H_2 * \dots * H_n)$$

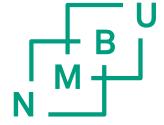
X/Y Separability



$$H_x = [1 \ 1 \ 1]$$

$$H_y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$(I * H_x) * H_y = I * (H_x * H_y)$$



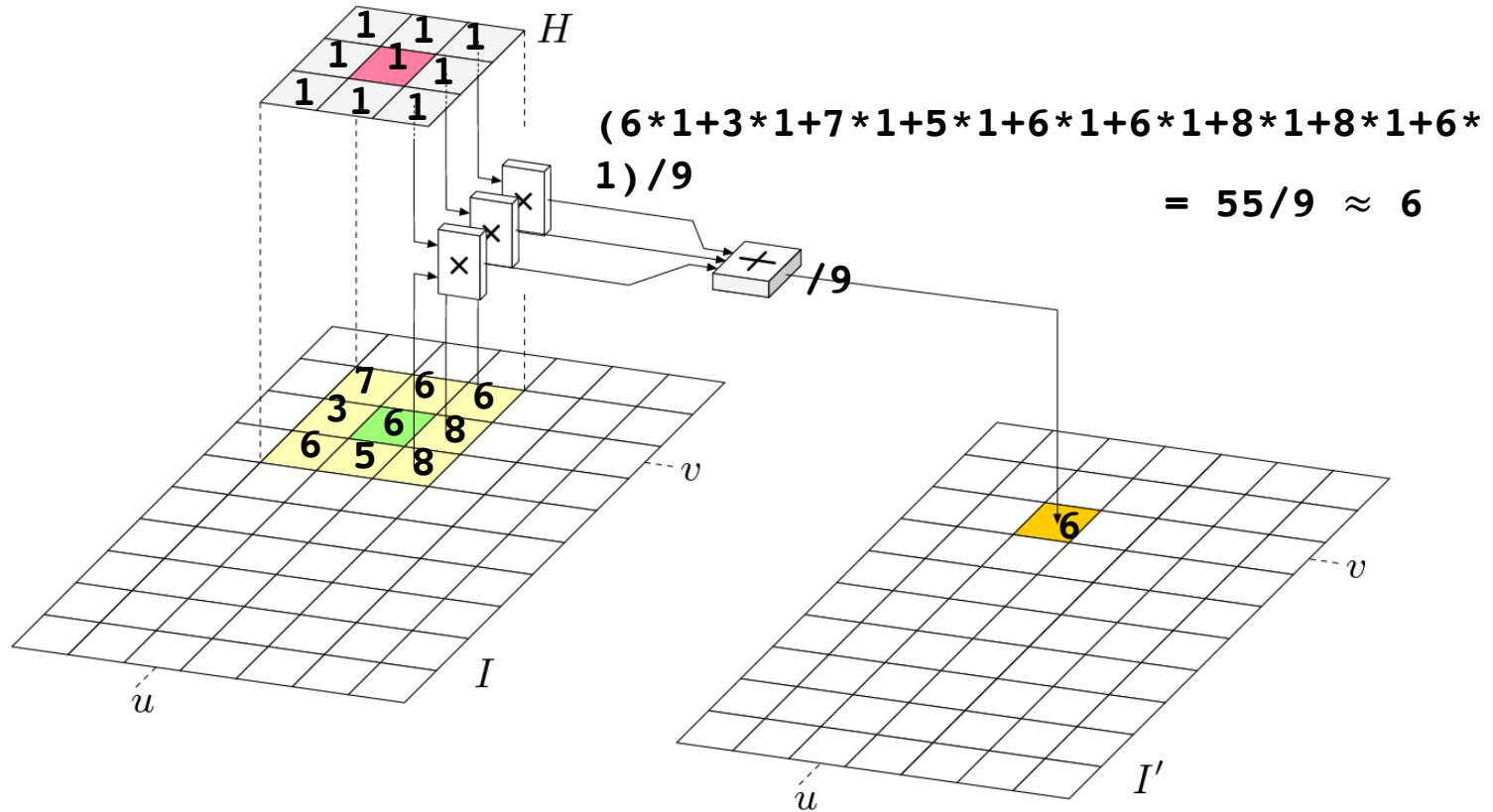
Smoothing mean filter

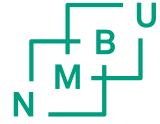
$$H_x = [1 \ 1 \ 1]$$

$$H_y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$H_{xy} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing mean filter



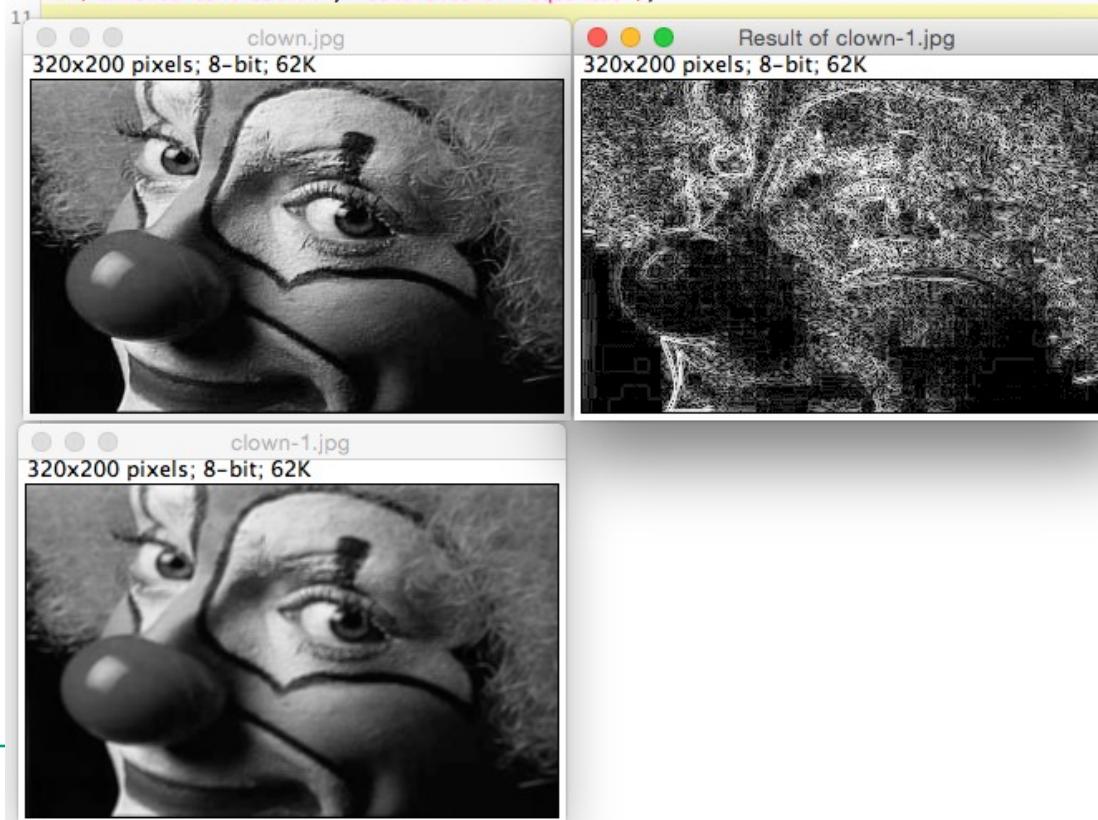


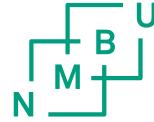
Analysing the residual image

- The residual image is the difference between the original image and the filtered image
- There can be a lot of information in a residual image

Smoothing and residual

```
1 // Macro som utfører glatting og finner residualbildet
2 // Residual = Original - Glattet (difference)
3 run("Clown (14K)");
4 run("8-bit");selectWindow("clown.jpg");
5 run("Duplicate...", " ");
6 run("Convolve...", "text1=[1 1 1\n1 1 1\n1 1 1] normalize");
7 selectWindow("clown-1.jpg");
8 imageCalculator("Difference create", "clown-1.jpg","clown.jpg");
9 selectWindow("Result of clown-1.jpg");
10 run("Enhance Contrast...", "saturated=0.4 equalize");
11
```

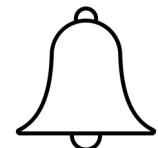




Other smoothing filters

- An alternative is to give the elements in $H(i,j)$ a bell form
- Focus is given to the pixels close the center of the filter (hot spot)

$$H(i, j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \mathbf{0.2} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}$$





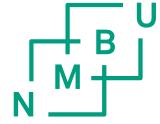
Integer coefficients

- Instead of using decimal numbers as coefficients in the filter matrix it is common to combine a new filter matrix with integer numbers and a constant

$$H(i, j) = s \cdot H'(i, j)$$

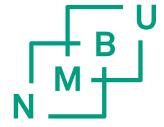
$$s = \frac{1}{\sum_{i,j} H'(i, j)}$$

$$H(i, j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix} = \frac{1}{40} \begin{bmatrix} 3 & 5 & 3 \\ 5 & \underline{8} & 5 \\ 3 & 5 & 3 \end{bmatrix}$$

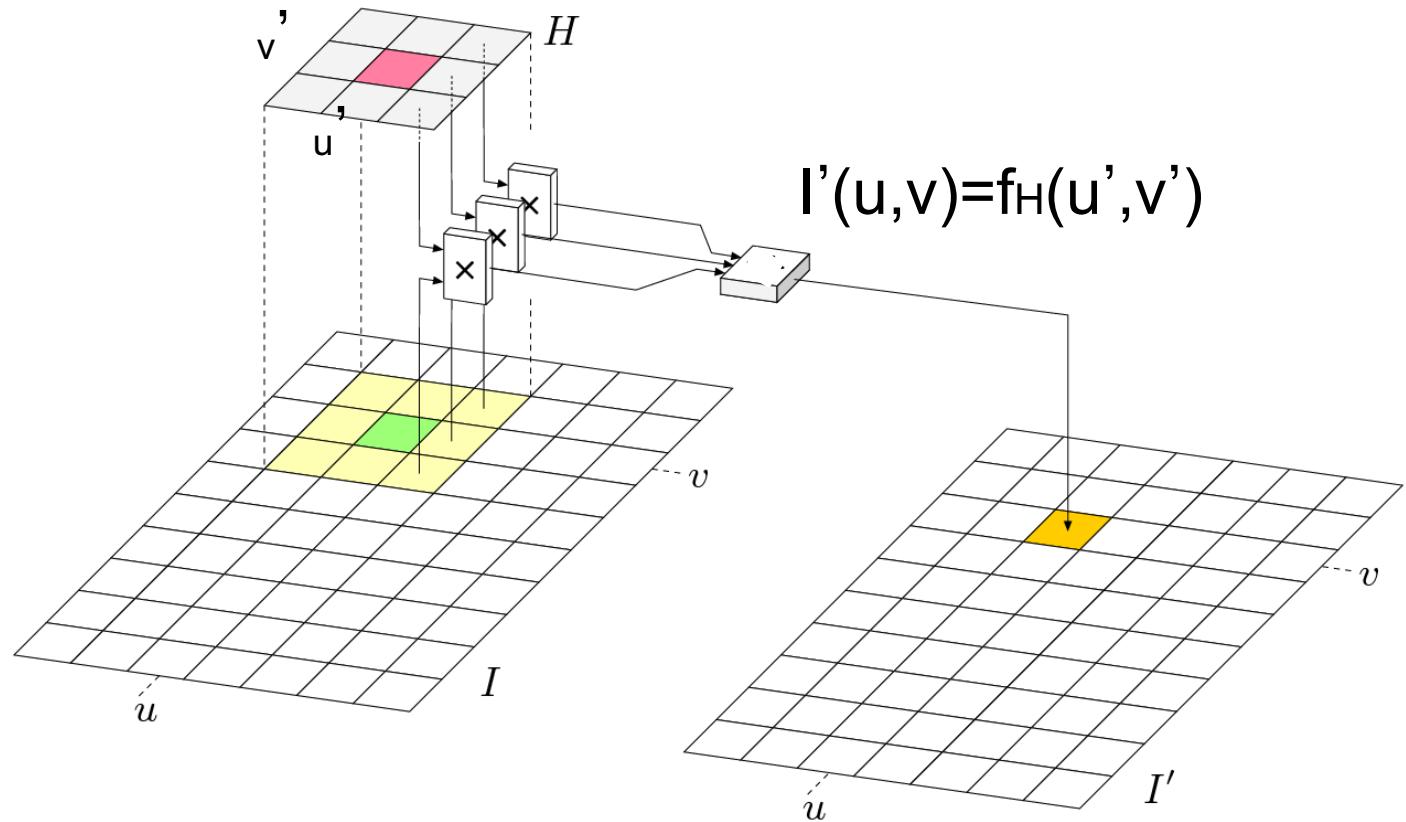


Nonlinear filters

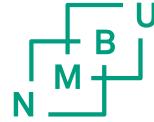
- Nonlinear filters also compute the result at some image position from the pixels inside a moving region of the original image.
- Nonlinear filters are combined by some nonlinear function
- Examples : maximum, minimum filters



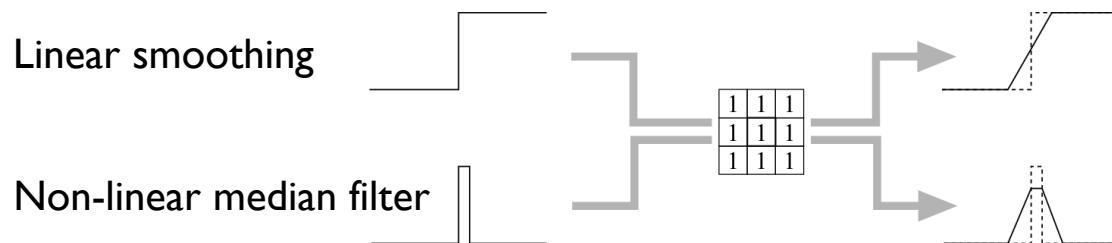
Nonlinear filters



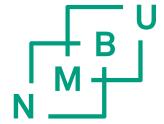
Nonlinear filters and noise reduction



- Linear smoothing filters have the inconvenience of smoothing edges and lines, reducing the image quality
- Nonlinear filters can be used as a better solution
- Mean vs median filter applied to noisy image -> ImageJ



Noise reduction



(a)
Original image

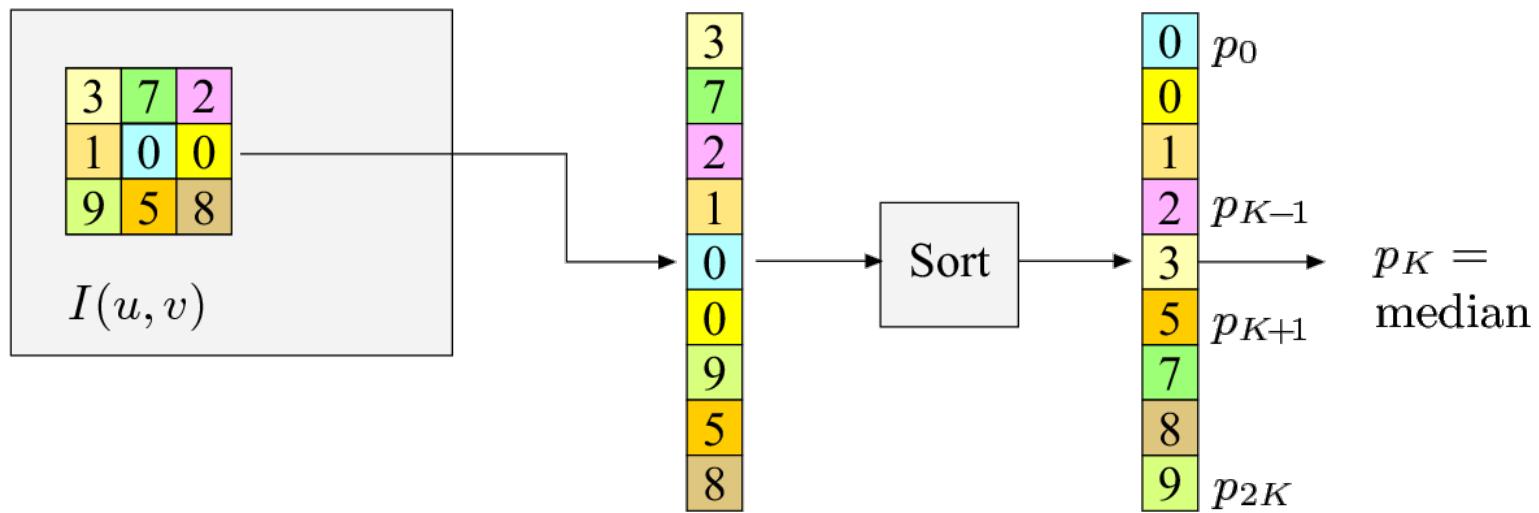
(b)
Minimum filter

(c)
Maximum filter

$$I'(u, v) \leftarrow \text{median} \{I(u+i, v+j) \mid (i, j) \in R\}$$

Median filter and noise reduction

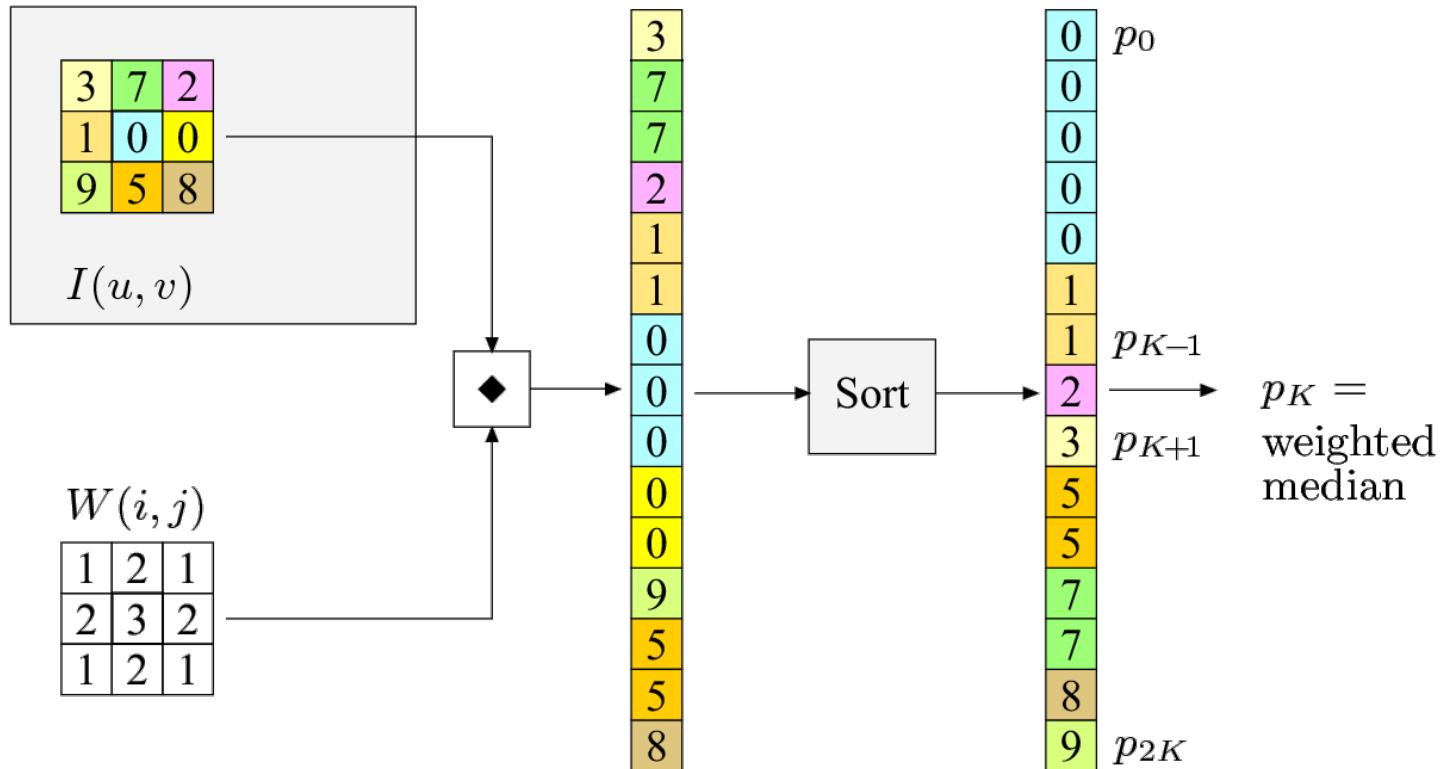
U
B
M
N



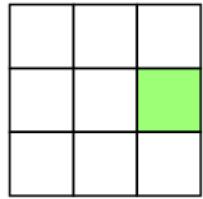
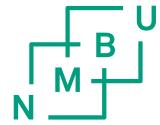
$$I'(u, v) \leftarrow \text{median} \{I(u+i, v+j) \mid (i, j) \in R\}$$

Weighted median filter and noise reduction

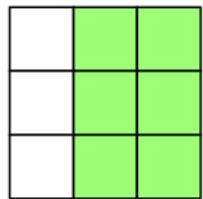
N M B U



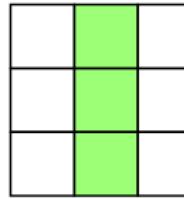
Median filter and noise reduction



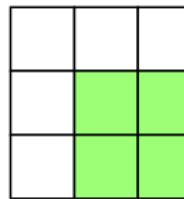
(a)



(c)



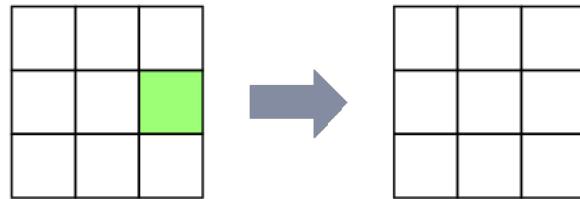
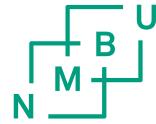
(b)



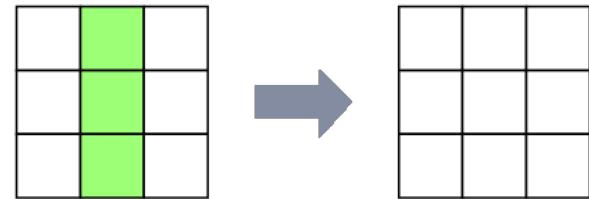
(d)



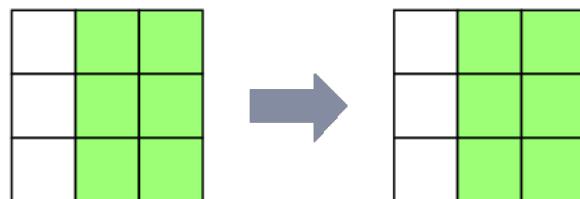
Median filter and noise reduction



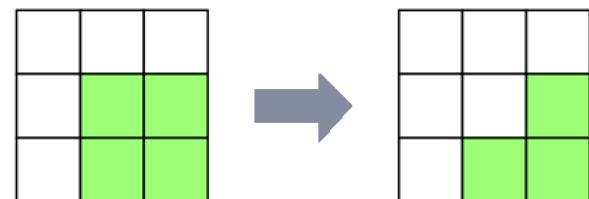
(a)



(b)



(c)



(d)

Figure 5.17 Effects of a 3×3 pixel median filter on two-dimensional image structures. Isolated dots are eliminated (a), as are thin lines (b). The step edge remains unchanged (c), while a corner is rounded off (d).

Borders

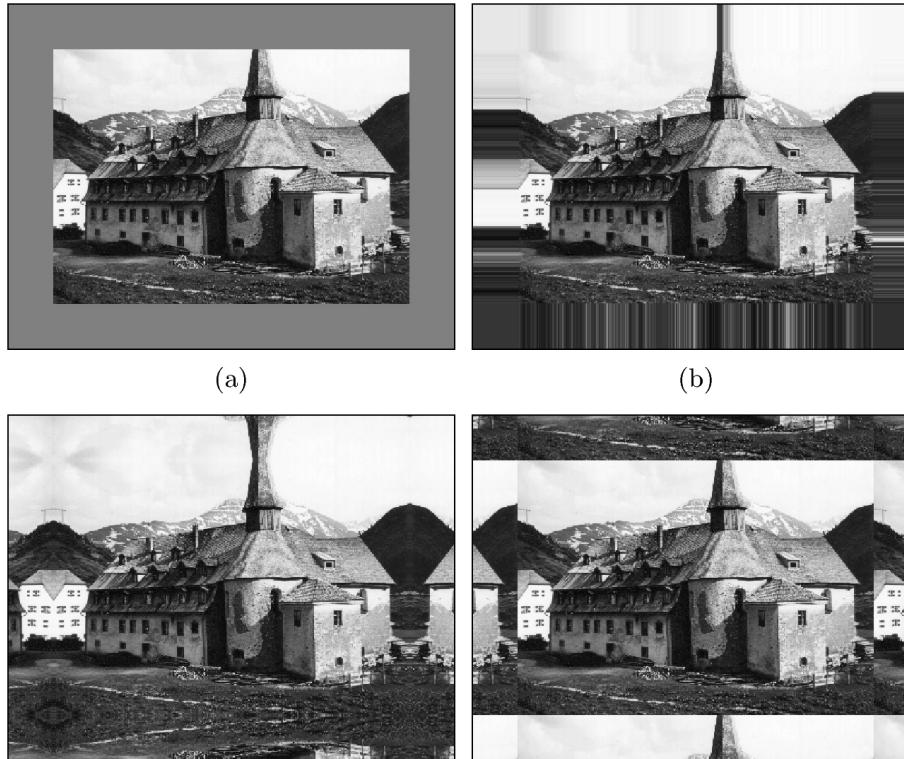
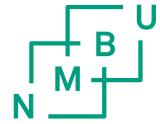
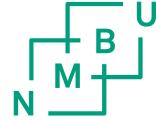
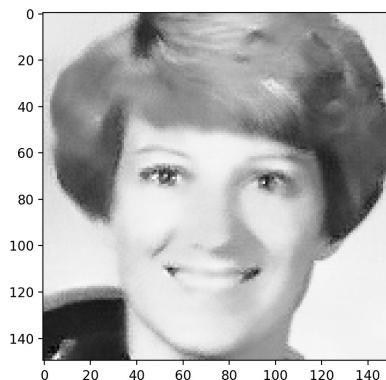
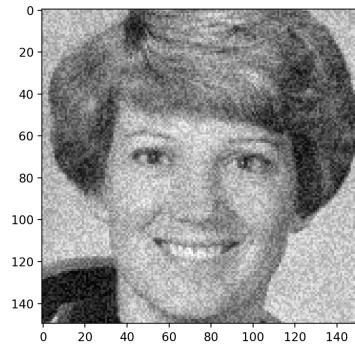


Figure 5.20 Methods for padding the image to facilitate filtering along the borders. The assumption is that the (nonexisting) pixels outside the original image are either set to some constant value (a), take on the value of the closest border pixel (b), are mirrored at the image boundaries (c), or repeat periodically along the coordinate axes (d).

Non local means



Better at preserving textures



```
import matplotlib.pyplot as plt
import numpy as np
from skimage.filters import median
from skimage.filters import gaussian
from skimage.morphology import disk
from skimage import data, img_as_float
from skimage.color import rgb2gray

from skimage.restoration import denoise_nl_means

astro1 = img_as_float(data.astronaut())
astro = rgb2gray(astro1)

astro = astro[30:180, 150:300]
plt.imshow(astro, 'gray')

noisy = astro + 0.3*np.random.random(astro.shape)
plt.imshow(noisy, 'gray')
noisy = np.clip(noisy, 0, 1)

medima = median(noisy,disk(2))
plt.imshow(medima, 'gray')

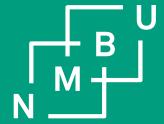
gaussima = gaussian(noisy, sigma=2)
plt.imshow(gaussima, 'gray')

denoise = denoise_nl_means(noisy, 7, 9, 0.08)
plt.imshow(denoise, 'gray')

difference = astro-denoise
plt.imshow(difference, 'gray')
```

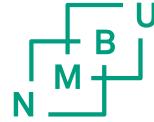
<https://www.youtube.com/watch?v=9tUns4HYtcw>

<https://www.youtube.com/watch?v=k8hS6uTz-Uc>



INF250

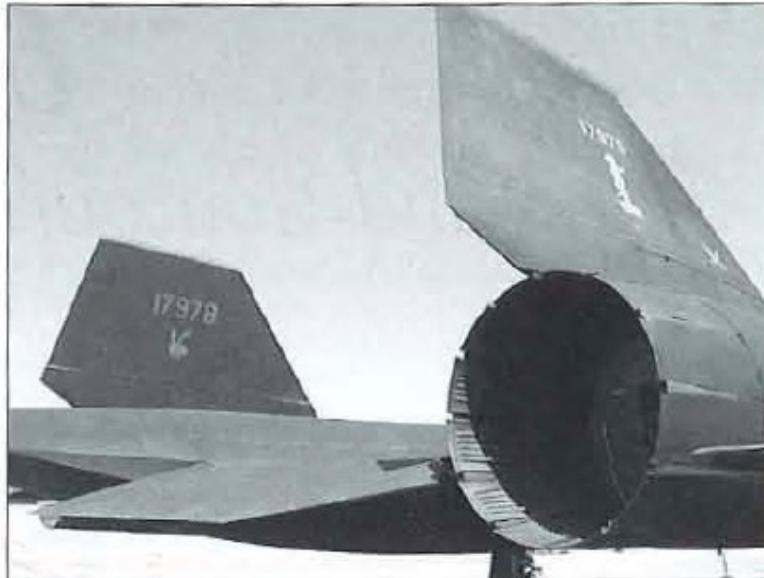
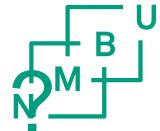
Filters: Edges and contours



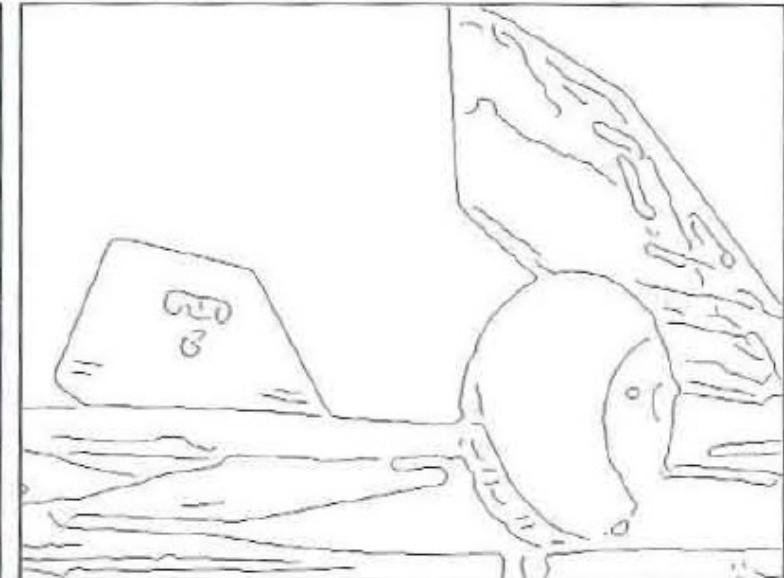
Edges and contours

- Edges and contours are important for humans and animals vision
- A contour of an object can easily be used to reconstruct the object itself
- An edge can be described as image positions where the local intensity vary dramatically in relation to its surroundings

How can edges and contours be localised



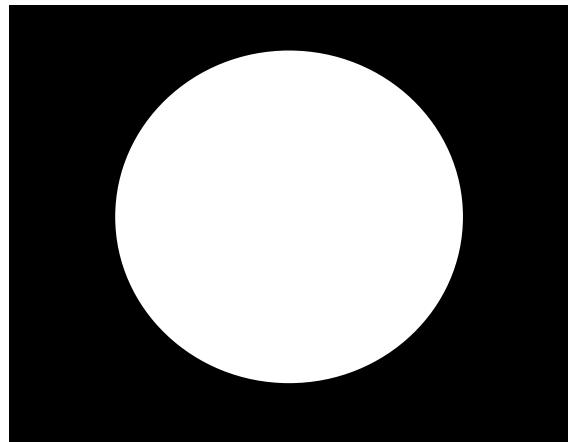
(a)



(b)

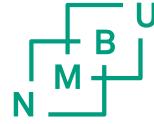
Figure 6.1 Edges play an important role in human vision. Original image (a) and edge image (b).

U
B
M
N



—

Gradient based edge detection



$$f'(x) = \frac{df}{dx}(x)$$

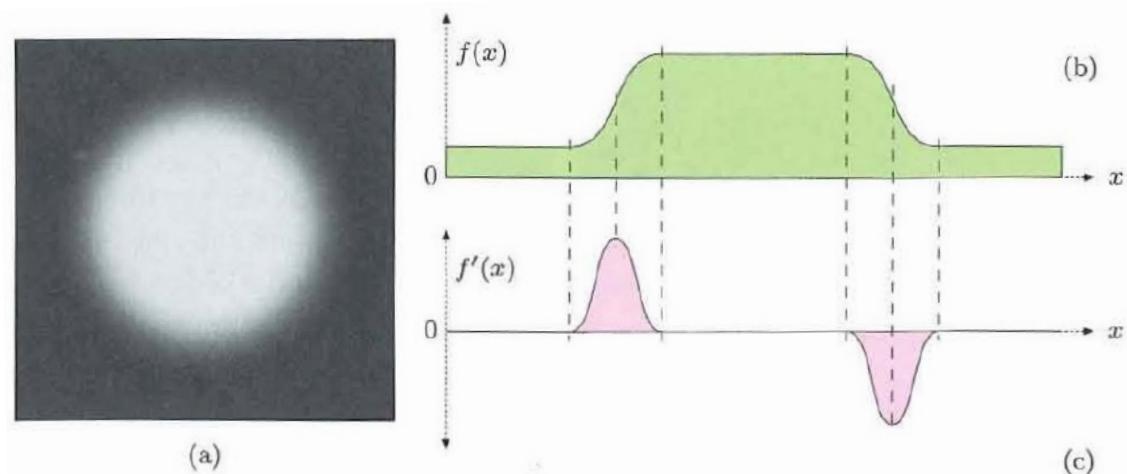


Figure 6.2 Sample image and first derivative in one dimension: original image (a), horizontal intensity profile $f(x)$ along the center image line (b), and first derivative $f'(x)$ (c).

Deriving a discrete signal

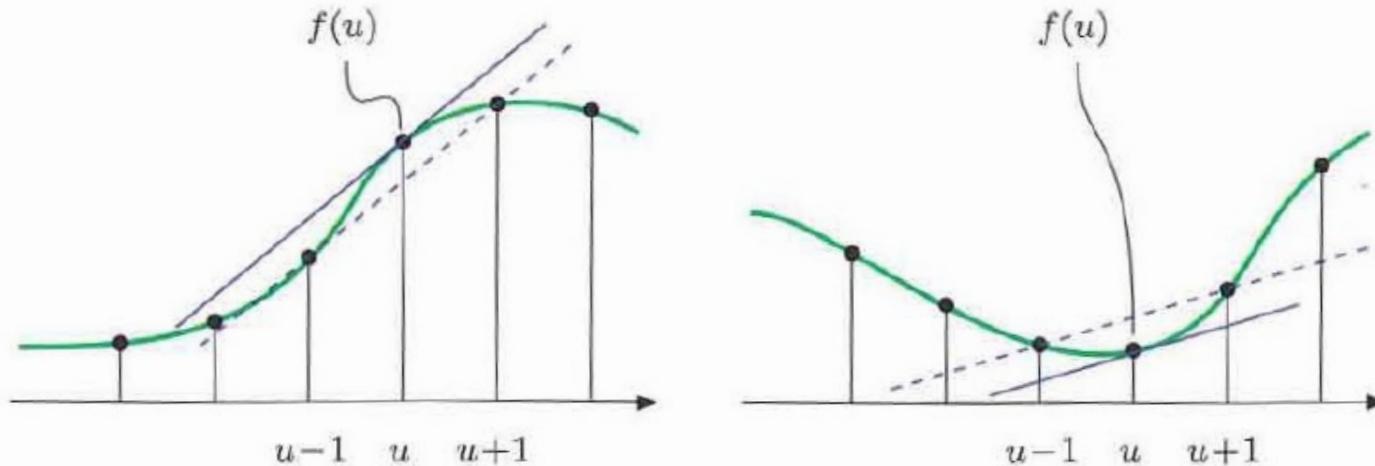
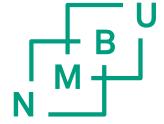


Figure 6.3 Estimating the first derivative of a discrete function. The slope of the straight (dashed) line between the neighboring function values $f(u-1)$ and $f(u+1)$ is taken as the estimate for the slope of the tangent (i.e., the first derivative) at $f(u)$.

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot (f(u+1) - f(u-1))$$



Partial derivation and gradient

$$\frac{\partial I}{\partial u}(u, v) \quad \text{and} \quad \frac{\partial I}{\partial v}(u, v)$$

$$\nabla I(u, v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u, v) \\ \frac{\partial I}{\partial v}(u, v) \end{bmatrix}$$

The magnitude of the gradient is invariant of image rotation, important for isotropic localization of edges.

$$|\nabla I|(u, v) = \sqrt{\left(\frac{\partial I}{\partial u}(u, v)\right)^2 + \left(\frac{\partial I}{\partial v}(u, v)\right)^2}$$

Simple edge operators

Linear filter and derivation

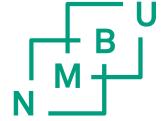


Horizontal and vertical component of the gradient can be obtained by the linear filters

$$H_x^D = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} = 0.5 \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$H_y^D = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Gradient filters

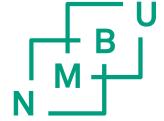


Prewitt operator

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_x^P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^* \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad H_y^P = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^* \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

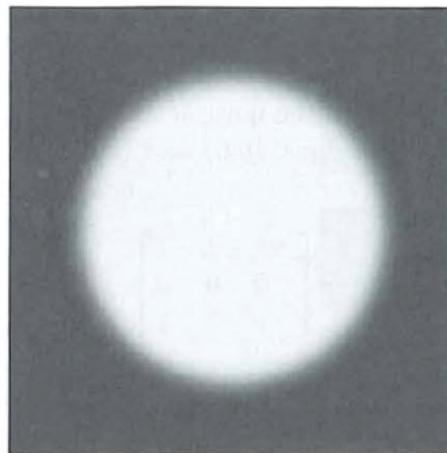
Gradient filters



Sobel operator – assigns higher weight to center line and column than the prewitt operator

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient - from 1 to 2 dimensions



I



I_x

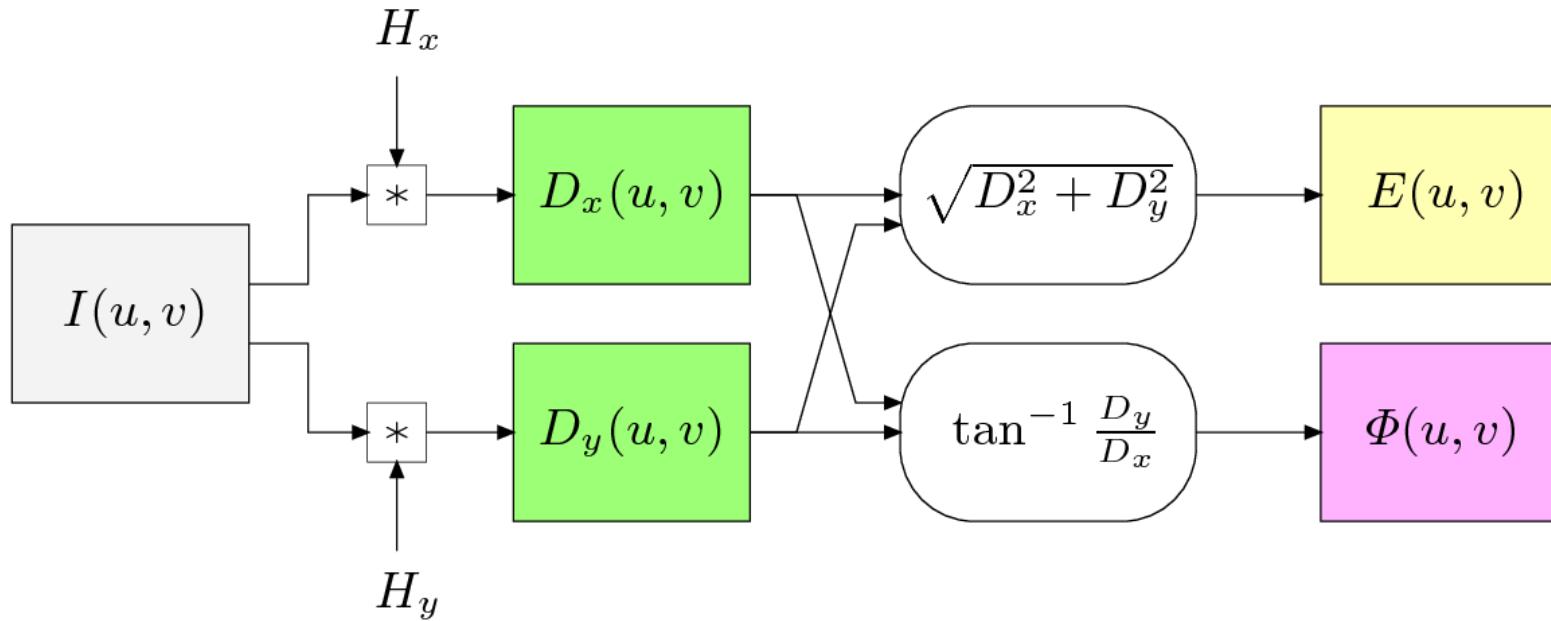
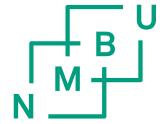


I_y

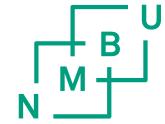


$|\nabla I|$

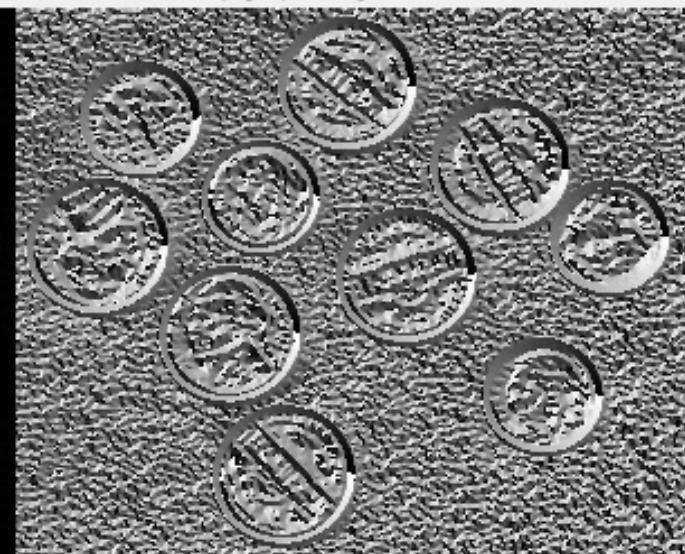
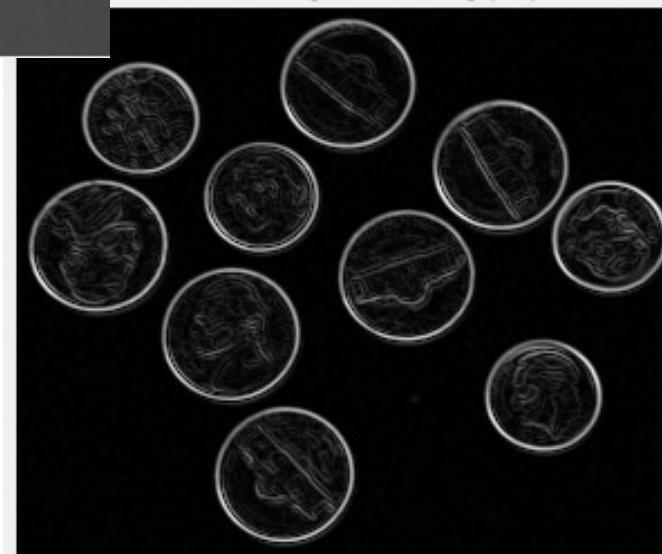
Gradientfiltere (SOBEL)



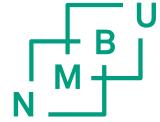
Gradientfilter, Orientation, Strength



Gradient Magnitude, Gmag (left), and Gradient Direction, Gdir (right), using Sobel method

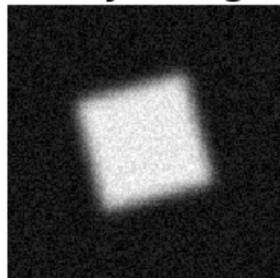


Canny filter

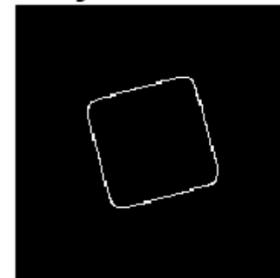
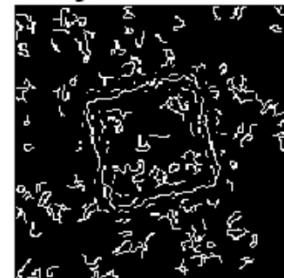


- Multi stage edge detector
- Based on the derivative of a Gaussian filtered image to compute the intensity of the gradients
- http://scikit-image.org/docs/dev/auto_examples/edges/plot_canny.html#sphx-glr-auto-examples-edges-plot-canny-py
- Two input parameters:
 - Sigma of gaussian smoothing
 - Hysteresis thresholding (min and max)

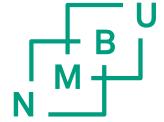
noisy image



Canny filter, $\sigma = 1$ Canny filter, $\sigma = 3$



Canny filter



1. Gaussian smoothing

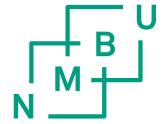
$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

2. Gradient calculation

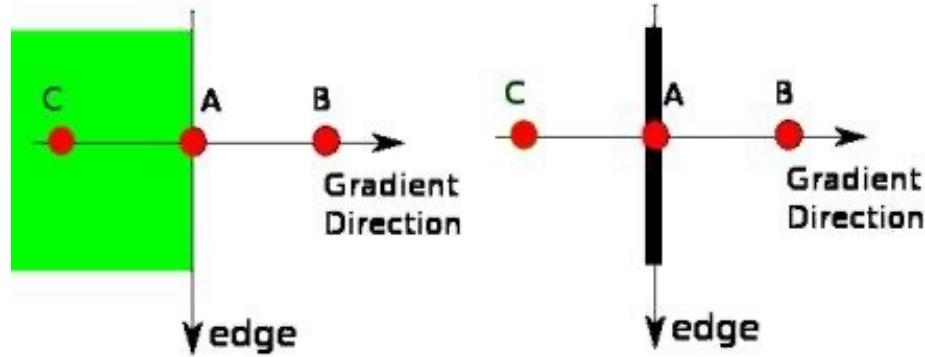
$$\text{Edge_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle } (\theta) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

Canny filter



3. Non-maximum suppression



Canny filter



3. Double thresholding

Strong edges:

$$E_{\text{strong}}(i, j) = \begin{cases} 1 & \text{if } M(i, j) \geq T_{\text{high}} \\ 0 & \text{otherwise} \end{cases}$$

Weak edges:

$$E_{\text{weak}}(i, j) = \begin{cases} 1 & \text{if } T_{\text{low}} \leq M(i, j) < T_{\text{high}} \\ 0 & \text{otherwise} \end{cases}$$

Non-edges:

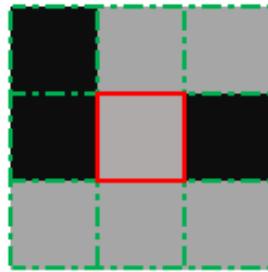
$$E_{\text{non-edge}}(i, j) = \begin{cases} 1 & \text{if } M(i, j) < T_{\text{low}} \\ 0 & \text{otherwise} \end{cases}$$

Canny filter

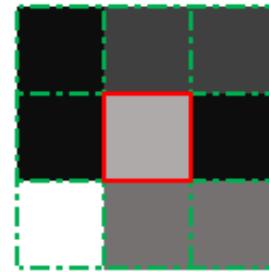


3. Edge tracking by hysteresis

- Strong edges kept
- Weak edges connected to a strong edge kept
- No-edge removed

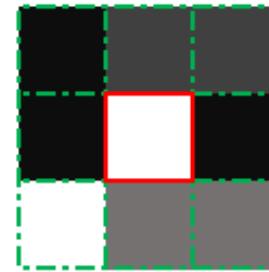


No strong pixels around

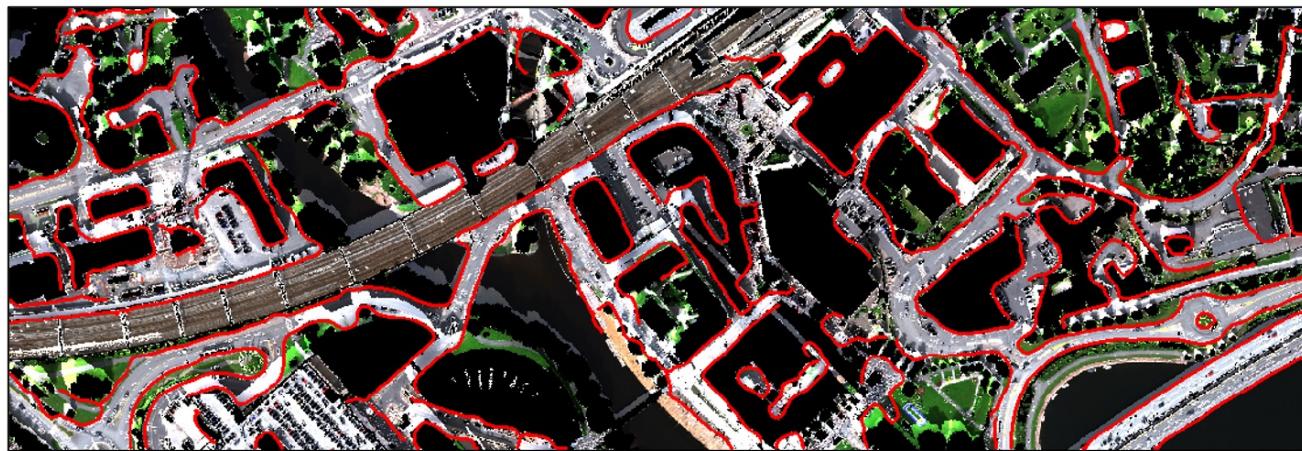
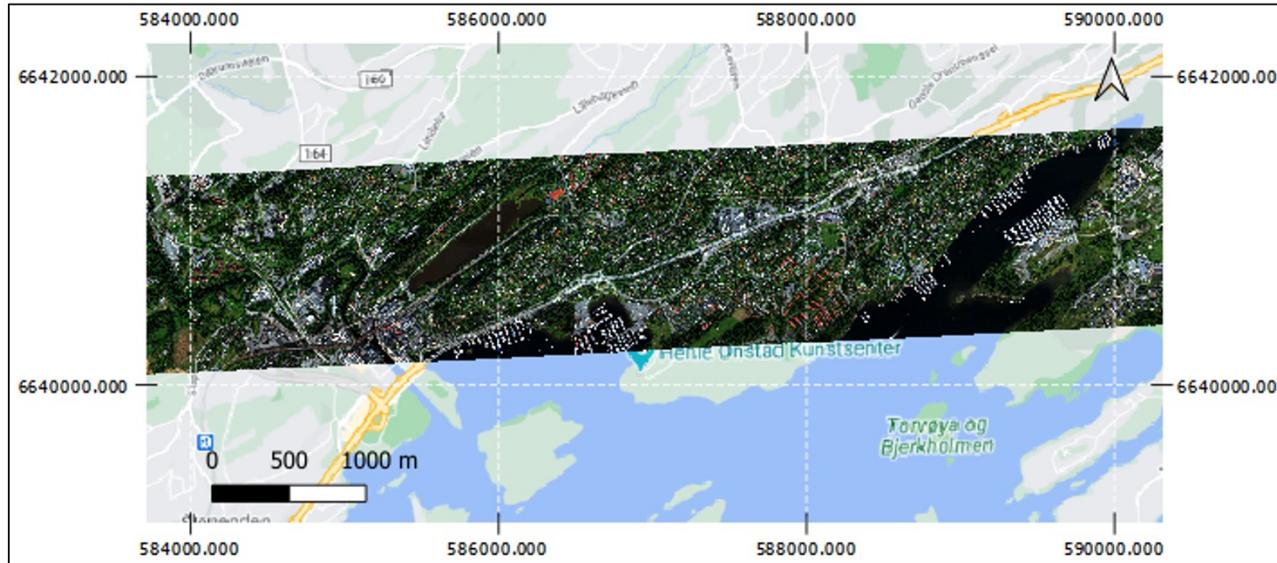


One strong pixel around

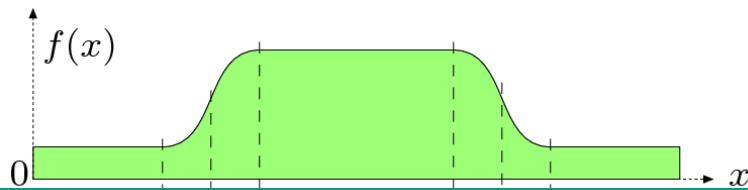
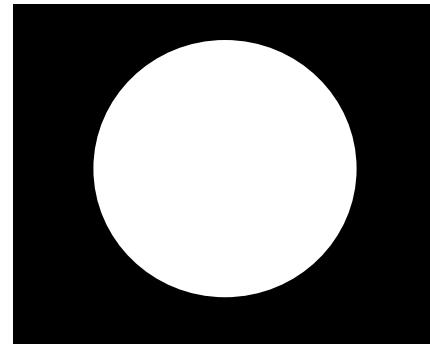
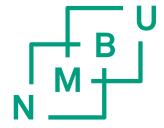
->



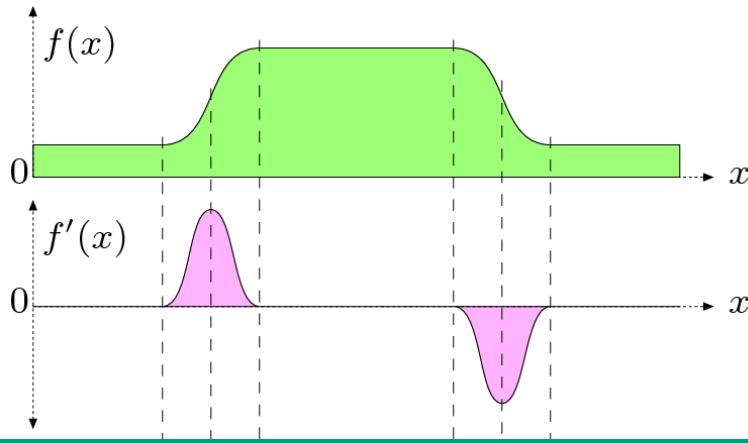
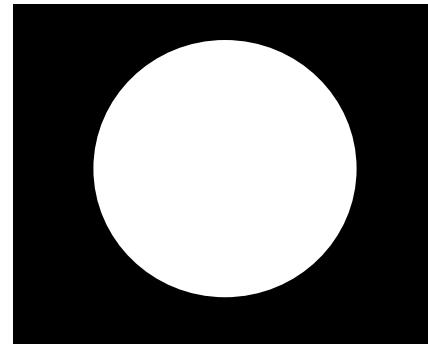
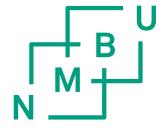
Canny filter for road detection

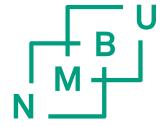


Laplace filter

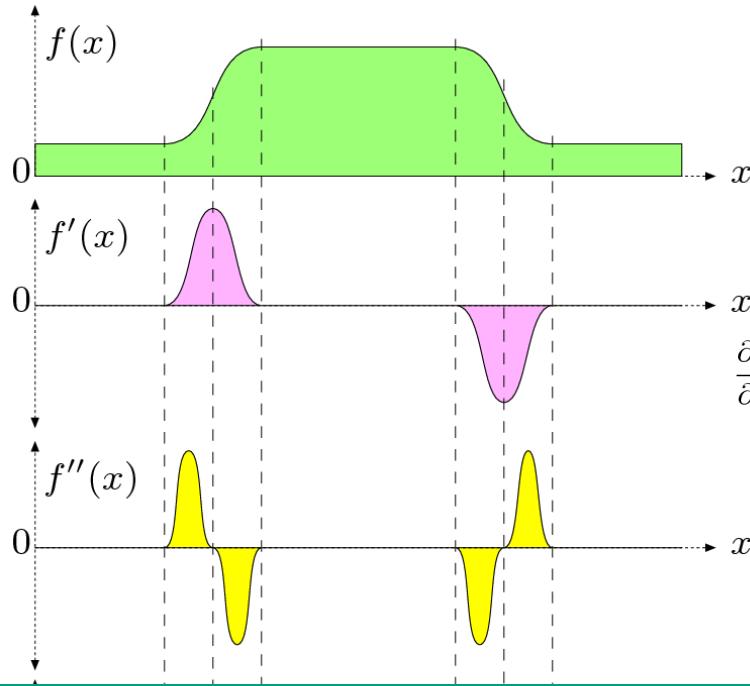
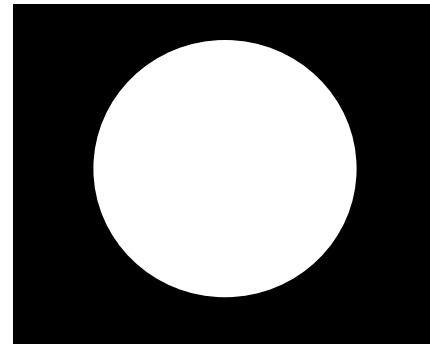


Laplace filter





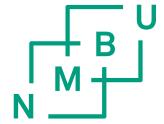
Laplace filter



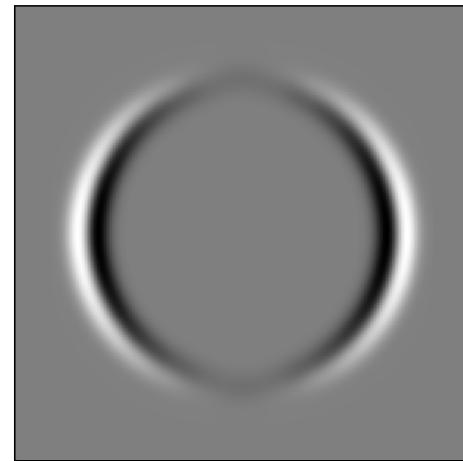
$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplace – edge sharpening



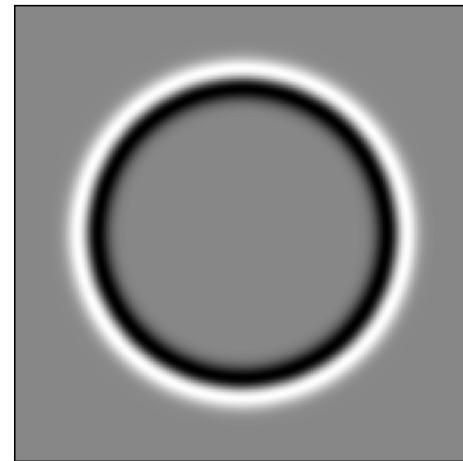
I



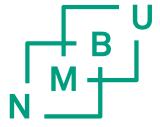
I_{xx}



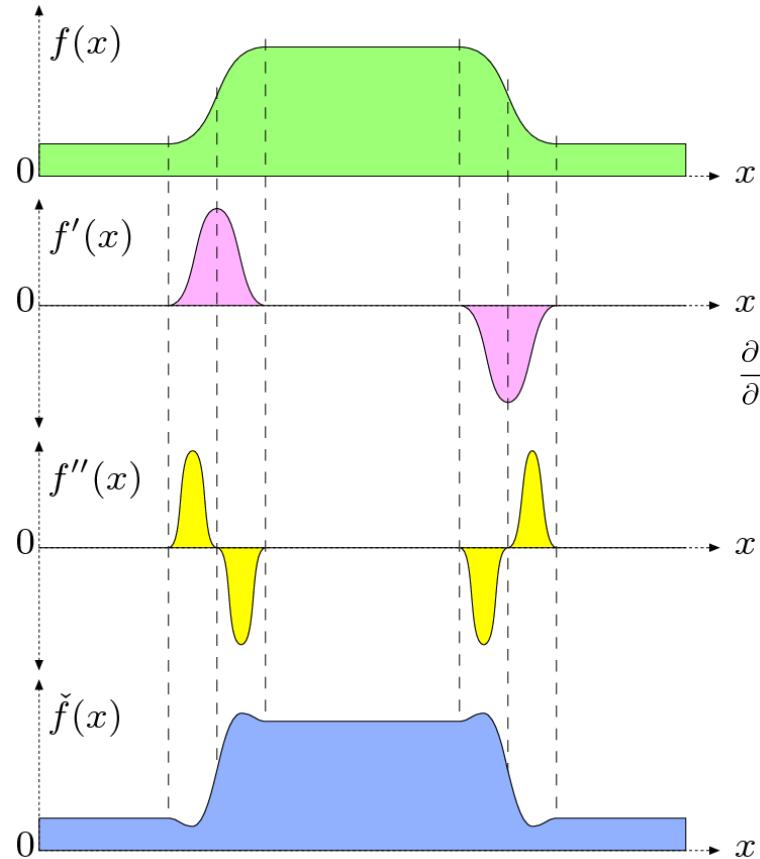
I_{yy}



$|\nabla^2 I|$

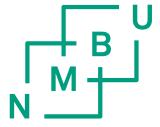


Laplace filter

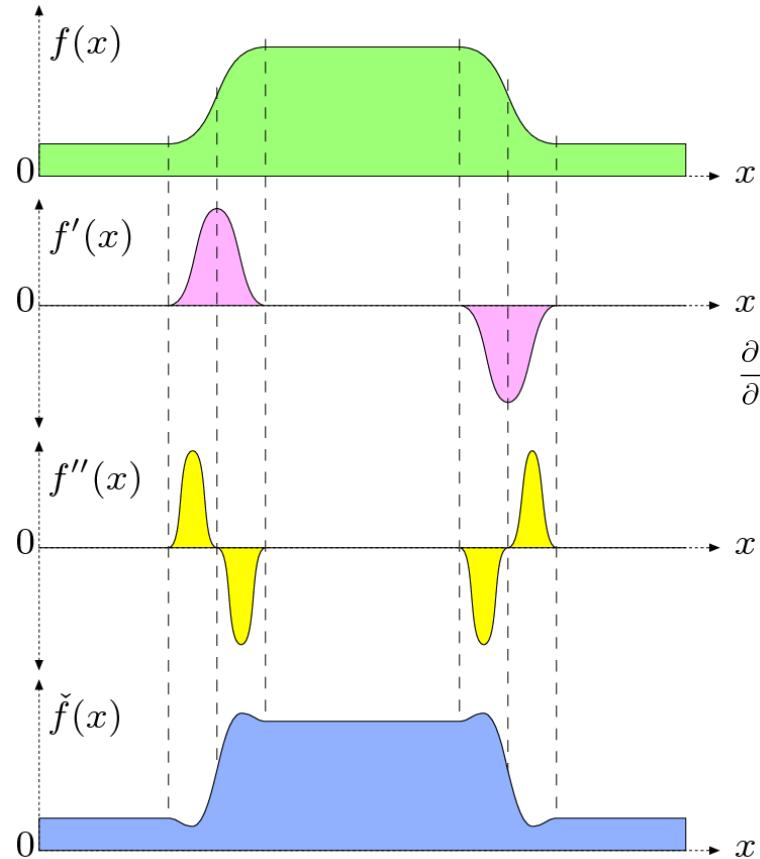


$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = [1 \ -2 \ 1] \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



Laplace filter

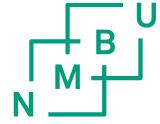


$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = [1 \ -2 \ 1]$$

$$\frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\tilde{f}(x) = f(x) - w \cdot f''(x)$$

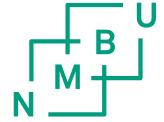


Low pass filter

- Filter \Leftrightarrow Convolution kernel
- Uniform weight convolution
 - The simplest filter (kernel)

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \text{"Lowpassfilter"}$$

Lowpass – passing low frequencies, coarse structures, long wavelengths



High pass filter

- Passing high frequencies / fine structures / short wavelengths
- High pass filtered image: Use a low pass filter and subtract result from the original image

$$H = I - L$$

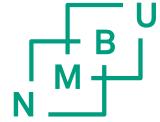
- H = high pass filtered image, I = original image, L = low pass filtered image

$$h \otimes I = i \otimes I - I \otimes I$$

- h , I and i are high pass, low pass and identity filter



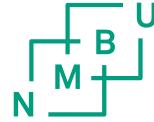
High pass filter cont.



$$i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \text{identity}$$

$$h = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \text{highpassfilter}$$

$$h = \frac{1}{9} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \text{highpassfilter}$$



Making the image sharper

- Sharpening kernel can be made by
 - Blending / mixing / overlaying an original image with a high pass filtered image (weighted)

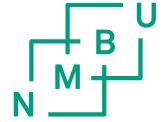
$$S = (1-f)*I + f * H$$

- S is the result, f is a factor ($0, 1$), I is the original image, H is the high pass filtered image

$$s \otimes I = (1-f)*i \otimes I + f*h \otimes I$$

$$s = (1-f)*i + f*h$$

Making the image sharper cont.



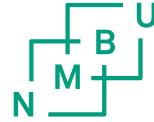
$$s = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + f \frac{1}{9} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Producing

$$s = \frac{1}{9} \begin{bmatrix} -f & -f & -f \\ -f & (9-f) & -f \\ -f & -f & -f \end{bmatrix}$$

$$S = s \otimes I$$

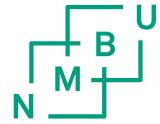
where s is the sharpening filter, I is the original image and S is the result



Unsharpen Mask (USM)

- USM is a technique to increase the sharpness using edge detections
- Popular within astronomy, digital printing, web publications etc
- USM was used in classical photography where sharp photos were constructed from a combination of the original photo and a smoothed (unsharp) photo
- The human vision uses the same principle

USM algorithm



- The mask M is generated by subtracting a smoothed version of the image from the original image (I)

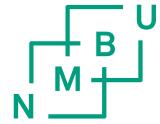
$$M \leftarrow I - (I * \tilde{H}) = I - \tilde{I}$$

where the kernel of the smoothing filter is assumed to normalized

- To obtain the sharpened image the mask is added to the original image, weighted by a factor a , which controls the amount of sharpening

$$\tilde{I} \leftarrow I + a \cdot M = I + a \cdot (I - \tilde{I}) = (1 + a) \cdot I - a \cdot \tilde{I}$$

Smoothing filter used in USM



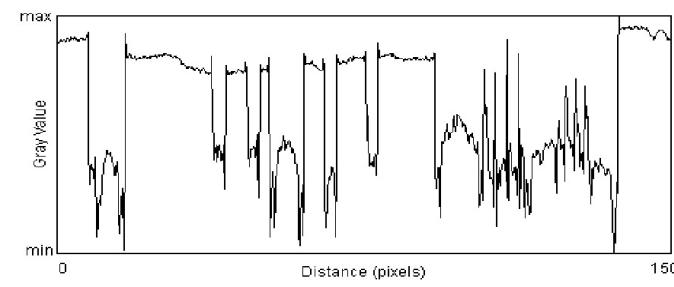
Any filter can be used but Gaussian filters with variable radius are most common



(a) Original



(b)



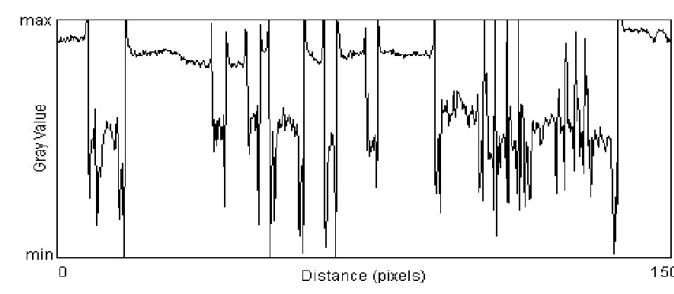
(c)



(d) $\sigma = 2.5$



(e)



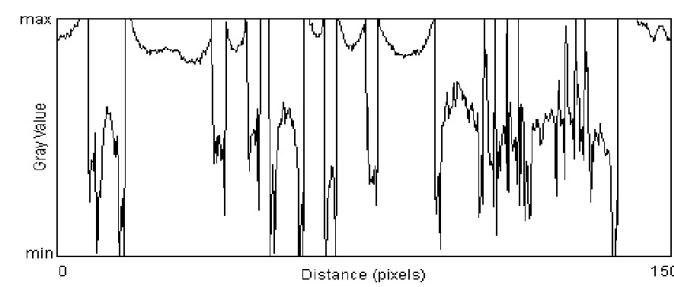
(f)



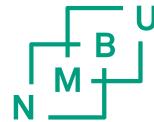
(g) $\sigma = 10.0$



(h)



(i)



```
import numpy as np
import matplotlib.pyplot as plt
from skimage import filters

from skimage import data, img_as_float
from skimage.data import astronaut
from skimage.color import rgb2gray

astro = rgb2gray(img_as_float(data.astronaut()))

astro = astro[30:180,150:300]
plt.imshow(astro,'gray')
```

```
#sobel and prewitt filter
astro_sobel = filters.sobel(astro)
plt.imshow(astro_sobel,'gray')

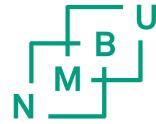
astro_sobel_h = filters.sobel_h(astro)
plt.imshow(astro_sobel_h,'gray')

astro_sobel_v = filters.sobel_v(astro)
plt.imshow(astro_sobel_v,'gray')

astro_prewitt = filters.prewitt(astro)
plt.imshow(astro_prewitt,'gray')

from skimage import feature
edges1 = feature.canny(astro, sigma=1)
plt.imshow(edges1,'gray')

edges2 = feature.canny(astro, sigma=3)
plt.imshow(edges2,'gray')
```



```
# laplace filter
from skimage.filters import laplace
astro_lap = laplace(astro)
plt.imshow(astro_lap, 'gray', vmin=0., vmax=0.2)
astro_sharp = astro-2*astro_lap
plt.imshow(astro_sharp, 'gray', vmin=0.4, vmax=0.9)
```

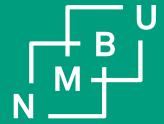
```
# high pass filter
from skimage.filters import gaussian
plt.imshow(astro, 'gray')
gaussastro = gaussian(astro, sigma=5)
plt.imshow(gaussastro, 'gray')
astro_high = astro-(gaussastro)
plt.imshow(astro_high, 'gray', vmin=0., vmax=0.2)

# unsharpening mask to sharpen the image
amount = 2
usm_astro = astro + amount*(astro-gaussastro)
plt.imshow(usm_astro, 'gray', vmin=0.2, vmax=0.9)

plt.imshow(astro, 'gray', vmin=0.2, vmax=0.9)

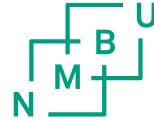
plt.hist(astro.ravel(), 256, [0,1], color='black')
plt.show()

plt.hist(gaussastro.ravel(), 256, [0,1], color='black')
plt.show()
```



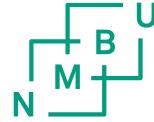
INF250

6 Morphology



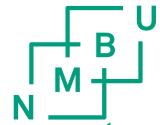
Morphology

- Morphological operations are used to modify the shape of an object, using local filter operations
- It can be used to remove unwanted effects in segmentation post-processing
 - Remove small objects (noise)
 - Smooth edges of larger objects
 - Fill in holes in objects
 - Link objects together
- It can be used as part of object description analysis
 - Locate boundary of objects
 - Thin objects
 - Locating objects within a certain structure
 - Locating patterns in an image
 - The operations are small and often very fast



Morphology and image types

- Morphology is used on binary images
- Filters can be used on gray level images
- Binary image
 - 1: Pixel in foreground (objects)
 - 0: Background
- Binary images are made through thresholding
- Morphology is about how to shrink or grow pixels of an object based on the location of the
- In this process a structure element is used as a filter



Median filter used as structuring element

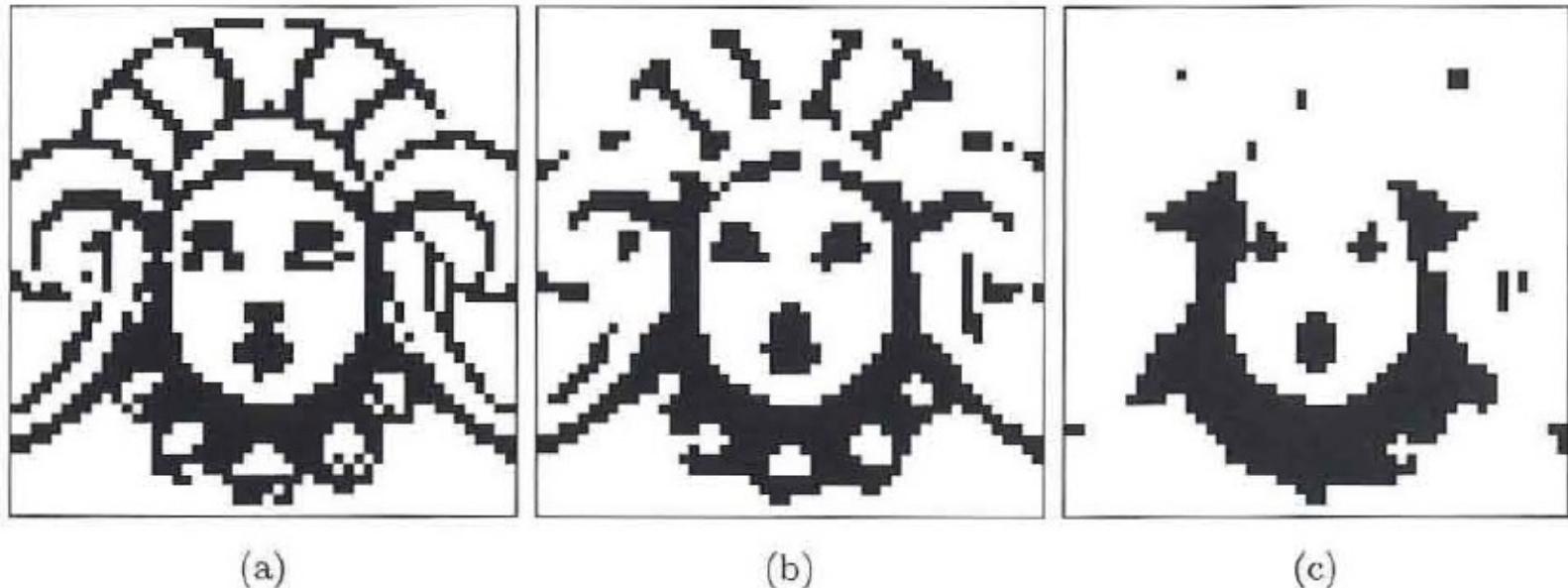
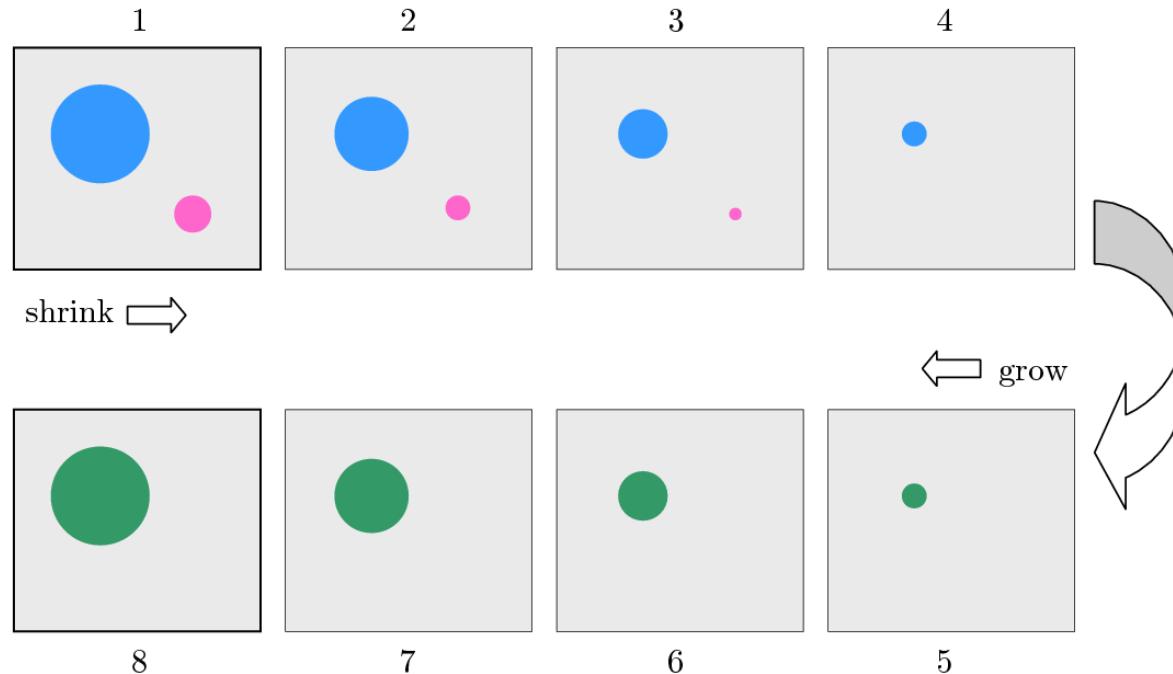
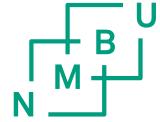


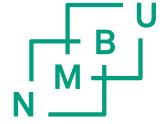
Figure 7.1 Median filter applied to a binary image: original image (a) and results from a 3×3 pixel median filter (b) and a 5×5 pixel median filter (c).

Shrink & Grow



- Can round off large structures and remove small structures
- Useful for preprocessing before further analysis

Neighborhood between pixels



\mathcal{N}_4

N_1 \times N_3
 N_2
 N_4

\mathcal{N}_8

N_5 N_2 N_6
 N_1 \times N_3
 N_8 N_4 N_7

Figure 7.5 Definitions of “neighborhood” on a rectangular pixel grid: 4-neighborhood $\mathcal{N}_4 = \{N_1, \dots, N_4\}$ (left) and 8-neighborhood $\mathcal{N}_8 = \mathcal{N}_4 \cup \{N_5, \dots, N_8\}$ (right).

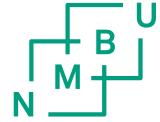
4 - neighborhood

8 - neighborhood



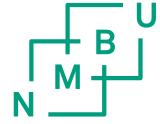
Morphological Operations

- In short: A set of operations that process images based on shapes. Morphological operations apply a *structuring element* to an input image and generate an output image.
- The most basic morphological operations are two: Erosion and Dilation.
 - Removing noise
 - Isolation of individual elements and joining disparate elements in an image.
 - Finding of intensity bumps or holes in an image



Dilation

- This operations consists of convoluting an image with some kernel (\cdot) , which can have any shape or size, usually a square or circle.
- The kernel has a defined *anchor point*, usually being the center of the kernel.
- As the kernel is scanned over the image, we compute the maximal pixel value overlapped by \cdot and replace the image pixel in the anchor point position with that maximal value.
- As you can deduce, this maximizing operation causes bright regions within an image to “grow” (therefore the name *dilation*).



Erosion

- This operation is the sister of dilation. What this does is to compute a local minimum over the area of the kernel.
- As the kernel is scanned over the image, we compute the minimal pixel value overlapped by and replace the image pixel under the anchor point with that minimal value.

Shrink and grow

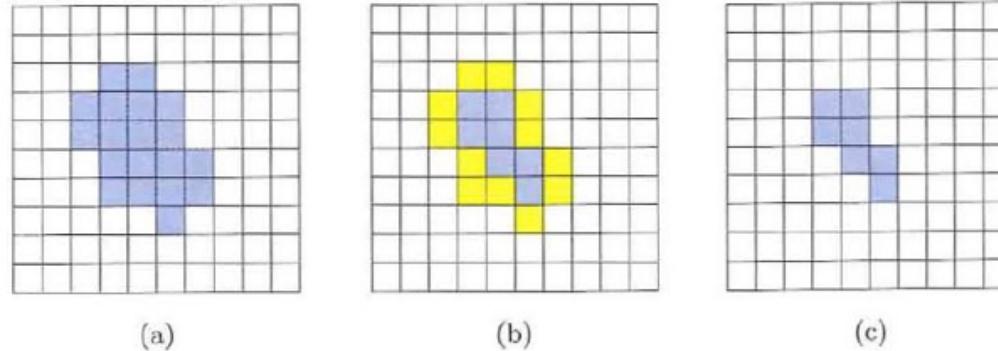
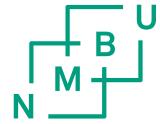


Figure 7.3 “Shrinking” a foreground region by removing a layer of border pixels: original image (a), identified foreground pixels that are in direct contact with the background (b), and result after shrinking (c).

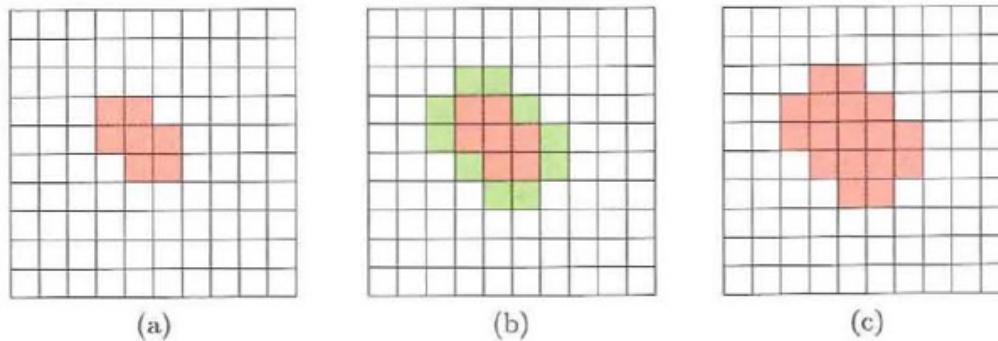


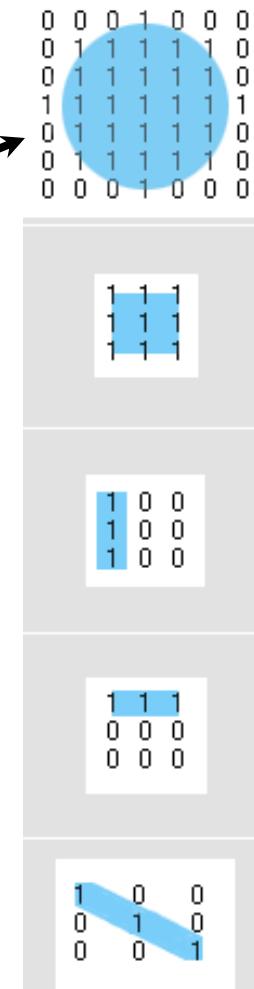
Figure 7.4 “Growing” a foreground region by attaching a layer of pixels: original image (a), identified background pixels that are in direct contact with the region (b), and result after growing (c).

Structuring element, SE (Morphological filter)



- A structuring element is applied
- Dilation with a circular SE with radius r adds a layer with thickness r to the foreground elements in the image
- Erosion with the same SE removes a layer of the same thickness

$r=3$

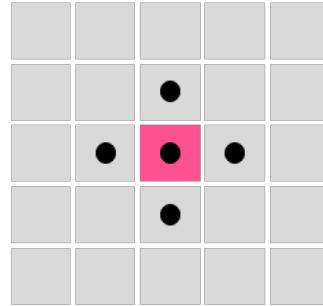


Structuring element (Morphology filter)

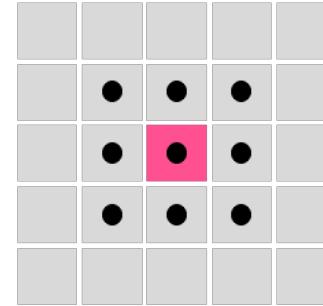


- A morphology filter is specified by
 - type of operation
 - Content of structuring element (SE)
- Size and form of SE is decided by
 - application
 - Form of object
 - structure in image, texture

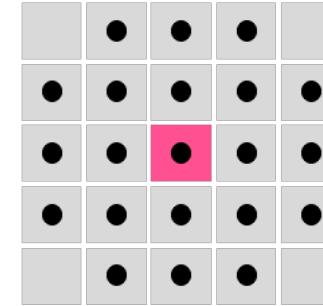
Examples of structuring elements - Neighborhood



(a)



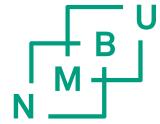
(b)



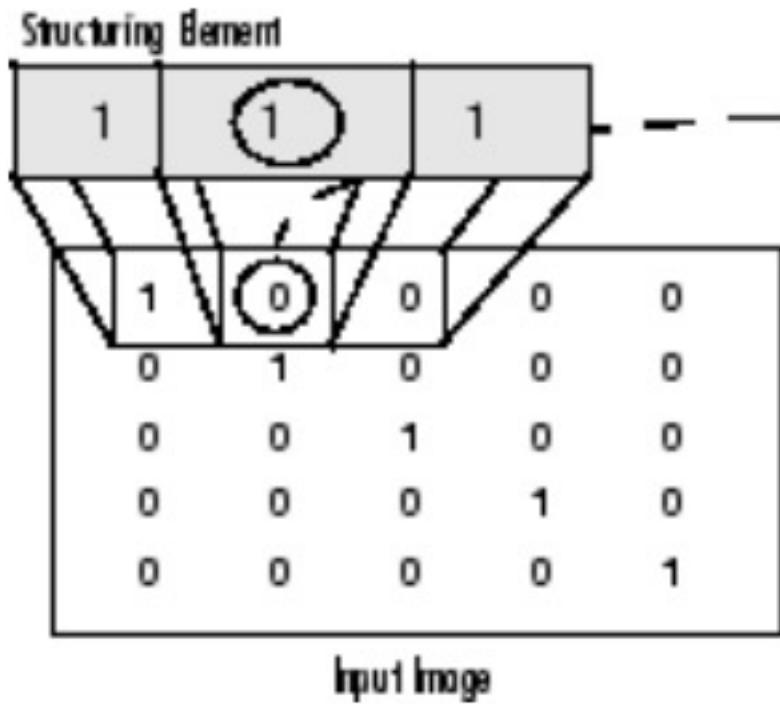
(c)

- (a) 4-neighborhood
- (b) 8-neighborhood
- (c) small circle with radius 2

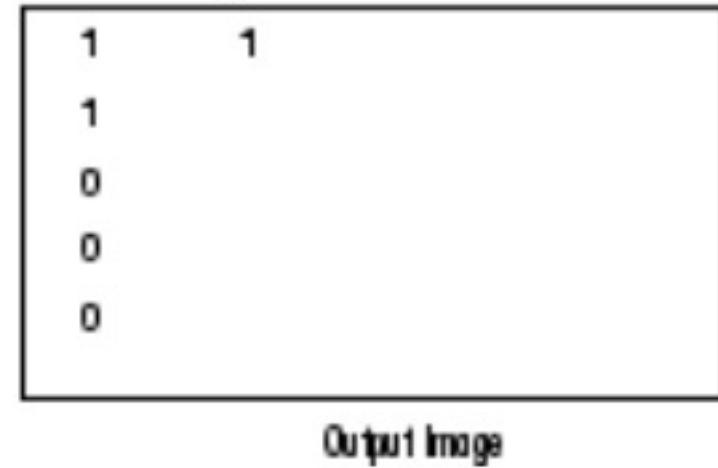
Graphical examples of dilation (maximum) (erosion is minimum)



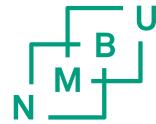
Morphological Dilation of a Binary Image



Maximum

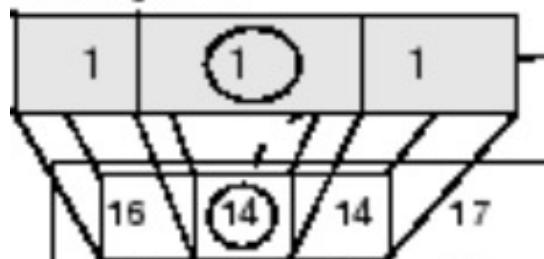


Graphical example of dilation (maximum) (erosion is minimum)



Morphological Dilation of a Grayscale Image

Structuring Element



Maximum

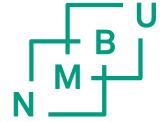
| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| 16 | 14 | 14 | 17 | 19 | 15 | 21 |
| 53 | 57 | 61 | 62 | 64 | 60 | 68 |
| 126 | 128 | 124 | 122 | 125 | 125 | 127 |
| 132 | 130 | 133 | 132 | 131 | 132 | 130 |
| 140 | 138 | 137 | 143 | 138 | 137 | 134 |
| 143 | 141 | 138 | 142 | 140 | 134 | 144 |
| 138 | 142 | 137 | 139 | 138 | 132 | 136 |

Input Image

| | |
| --- | --- |
| 16 | 16 |
| 57 | |
| 128 | |
| 132 | |
| 140 | |
| 143 | |
| 142 | |

Output Image

Gray level morphology



- The structuring element is not only binary
- values $H(i, j) \in \mathbb{R}$, for $(i, j) \in \mathbb{Z}^2$
- Dilation $(I \oplus H)(u, v) = \max_{(i, j) \in H} \{I(u+i, v+j) + H(i, j)\}$
- Erosion $(I \ominus H)(u, v) = \min_{(i, j) \in H} \{I(u+i, v+j) - H(i, j)\}$

Gray level morphology

- Burger and Burge describes gray level dilation (erosion) by summing (subtracting) the structuring element with corresponding pixels and replace the center pixel with max (min) value

$$\begin{array}{c}
 I \\
 \oplus \\
 H
 \end{array}
 = I \oplus H$$

$$I + H$$

$$I \ominus H = I \ominus H$$

$$I - H$$

Dilation and erosion



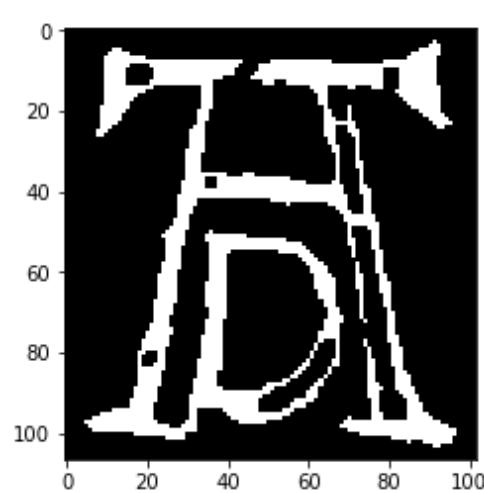
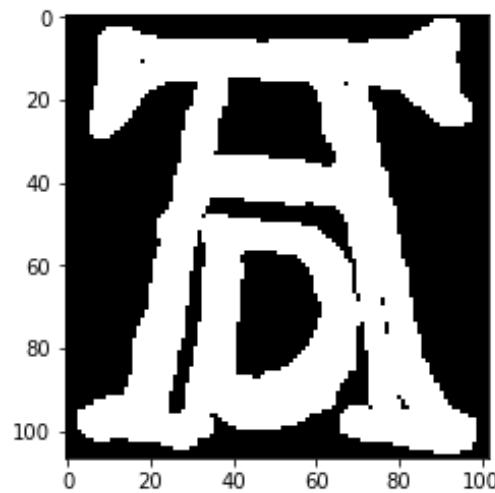
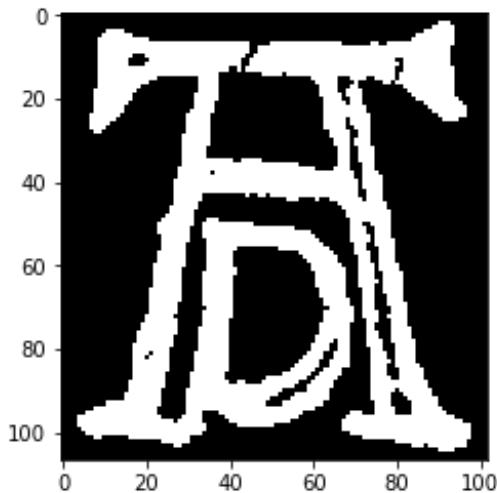
```
from skimage.morphology import skeletonize, dilation, opening, square
from skimage.morphology import erosion, closing
from skimage import io

filename = 'rhino_detail.tif'

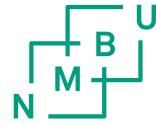
rhino = io.imread(filename)
plt.imshow(rhino,'gray')

rhinodil = dilation(rhino,square(3))
plt.imshow(rhinodil,'gray')

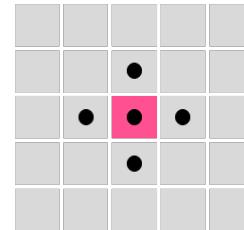
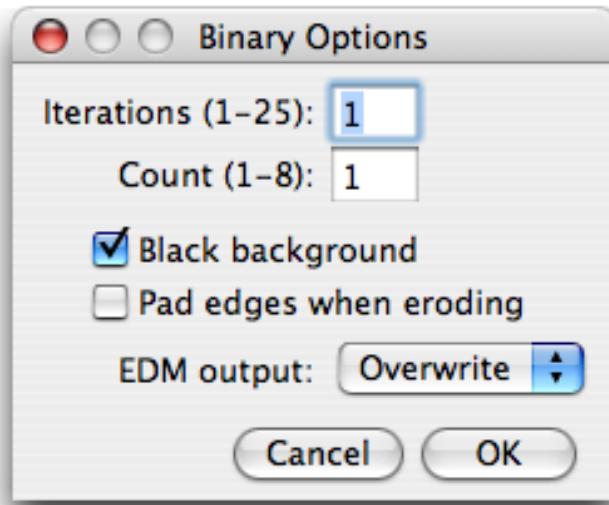
rhinoerode = erosion(rhino,square(3))
plt.imshow(rhinoerode,'gray')
```



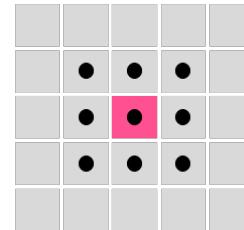
ImageJ



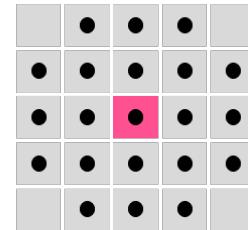
- ImageJ has implemented a 8-neighborhood structuring element (b).



(a)

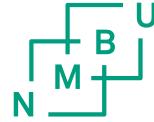


(b)



(c)

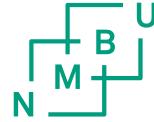




ImageJ binary menu

- **Convert to Mask**
- Converts the image to black and white based on the current threshold settings (if set) or on a threshold calculated by analyzing the histogram. The mask will have an inverting LUT (white is 0 and black is 255) unless "Black Background" is checked in the *Process>Binary>Options* dialog box.
- **Erode**
- Removes pixels from the edges of black objects. Use *Process>Filters>Minimum* to do grayscale erosion.
- **Dilate**
- Adds pixels to the edges of black objects. Use *Process>Filters>Maximum* to do grayscale dilation.
- **Open**
- Performs an erosion operation, followed by dilation. This smoothes objects and removes isolated pixels.
- **Close**
- Performs a dilation operation, followed by erosion. This smoothes objects and fills in small holes.

Example



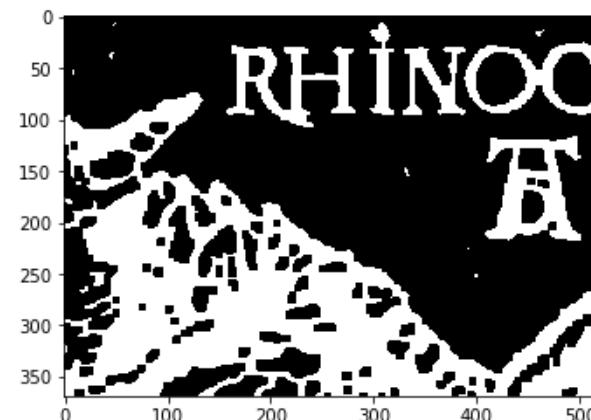
dilation



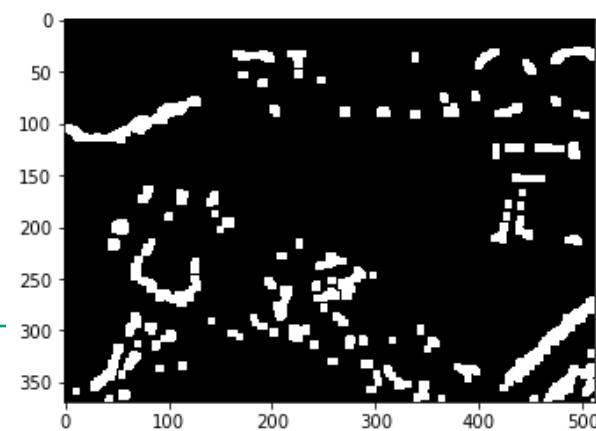
erosion



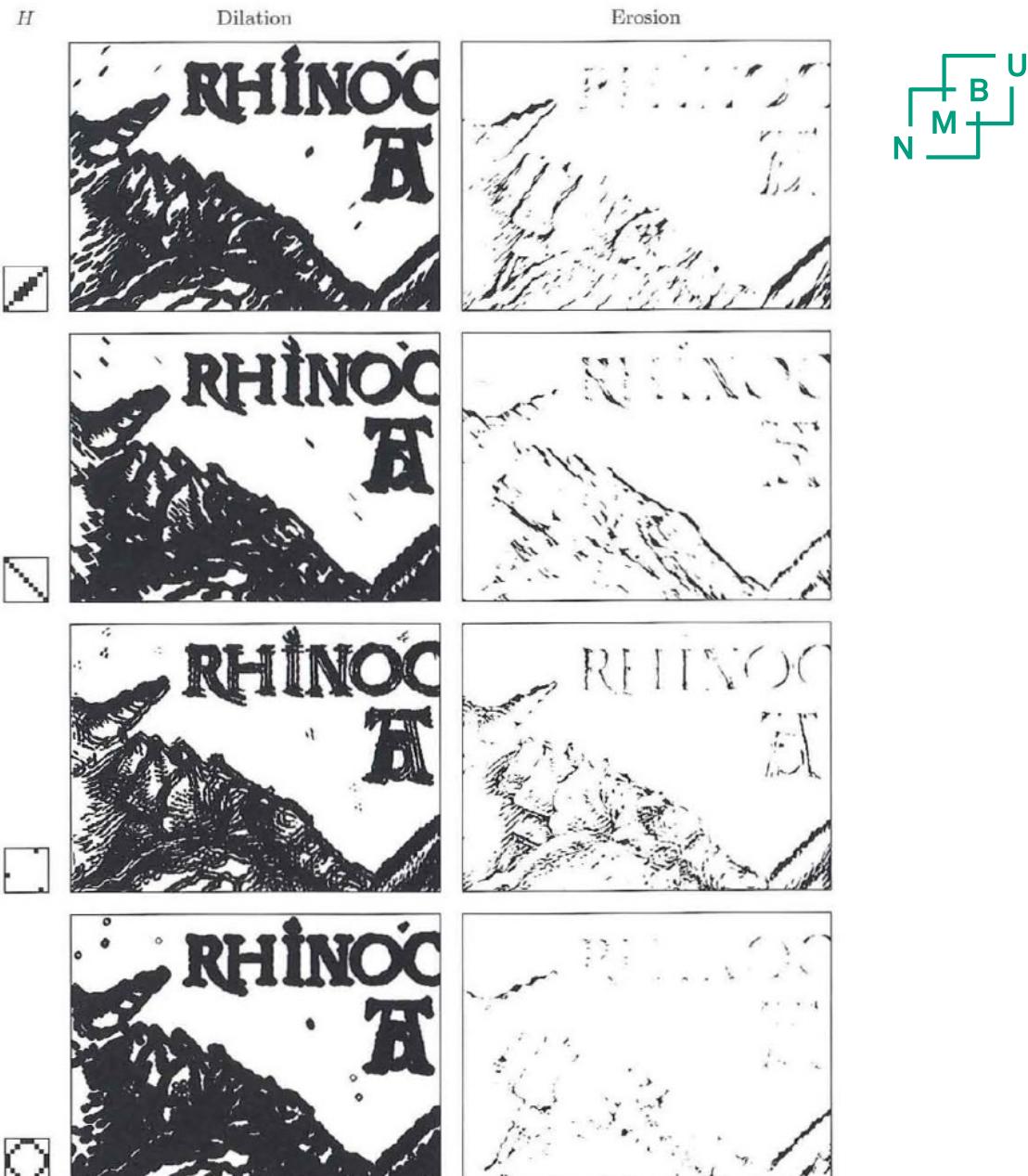
Square 4



Square 6



Example

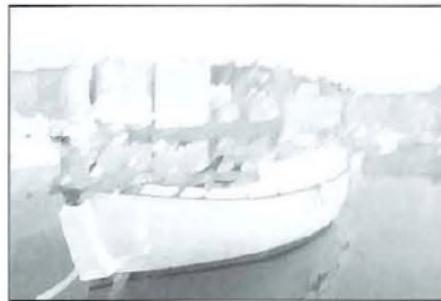


Example

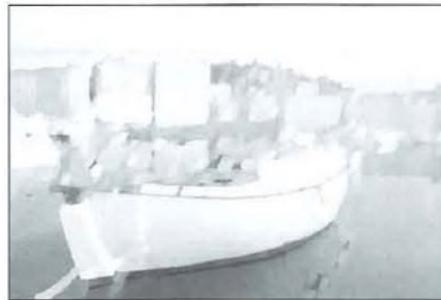
H



Dilation

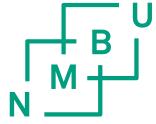


Erosion



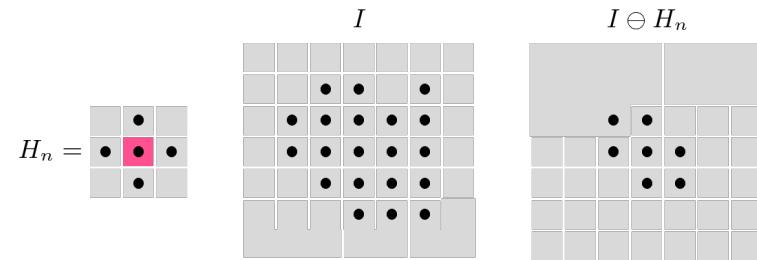
Morfologi

Figure 7.22 Grayscale dilation and erosion with various free-form structuring elements.



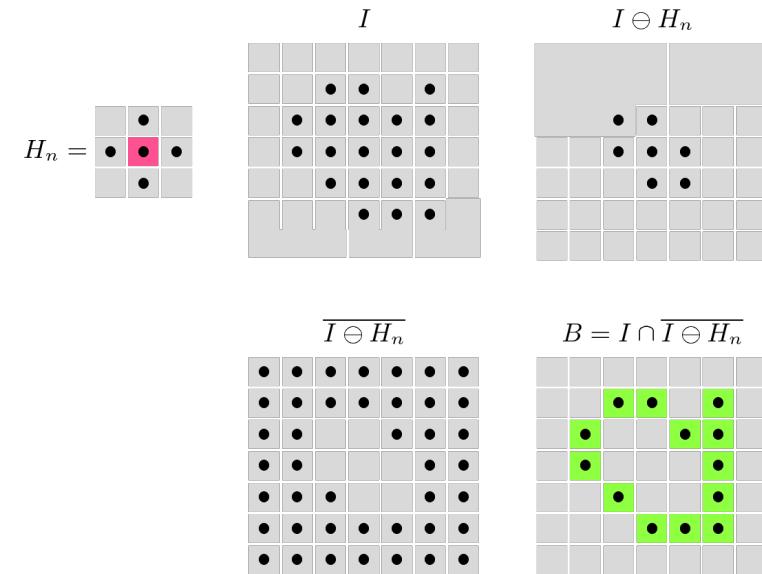
Outlining

- Dilation and erosion are combined with mathematical operations
- Example: contours, outlining
- Neighborhood kept (different from Sobel/Prewitt filter)

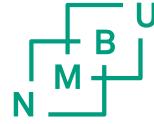


Outlining

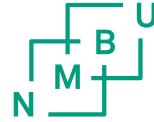
- Dilation and erosion are combined with mathematical operations
- Example: contours, outlining
- Neighborhood kept (different from Sobel/Prewitt filter)



Opening

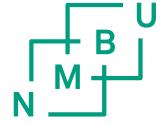


- Opening: $I \circ H = (I \ominus H) \oplus H$
- A binary opening is defined by erosion followed by a dilation
- Main effect: all foreground structures smaller than the structuring element will be eliminated in the erosion. The leftover structures will be smoothed in the following dilation
- In other words:
 - Shrinking followed by growth will eliminate the small structures



Closing

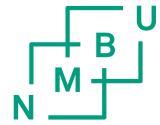
- Closing: $I \bullet H = (I \oplus H) \ominus H$
- A binary closing is defined by a dilation followed by an erosion
- Main effect: all holes and “impurities” smaller than the structuring element will be eliminated
- In other words:
 - Growth followed by shrinking will eliminate holes and small structures



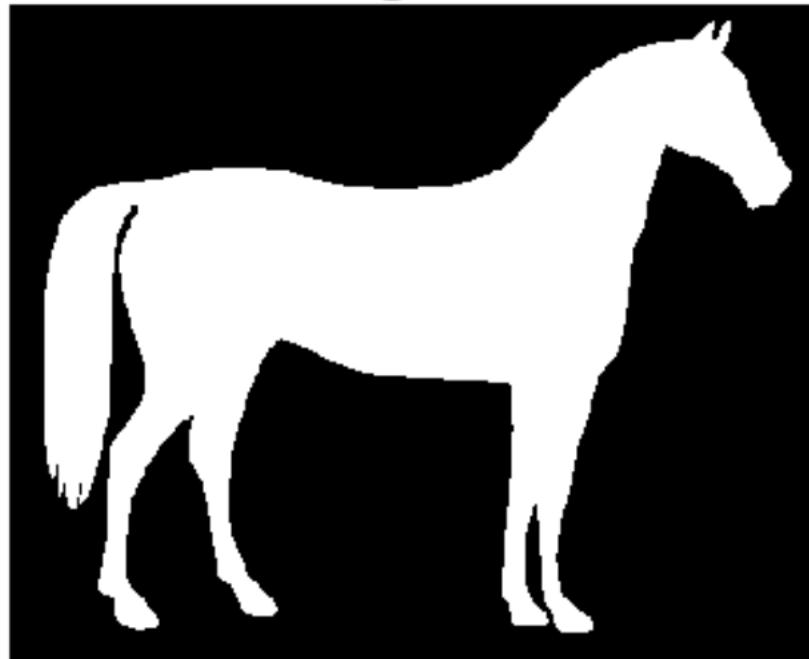
python

<http://scikit-image.org/docs/dev/api/skimage.morphology.html?highlight=dilation#skimage.morphology.dilation>

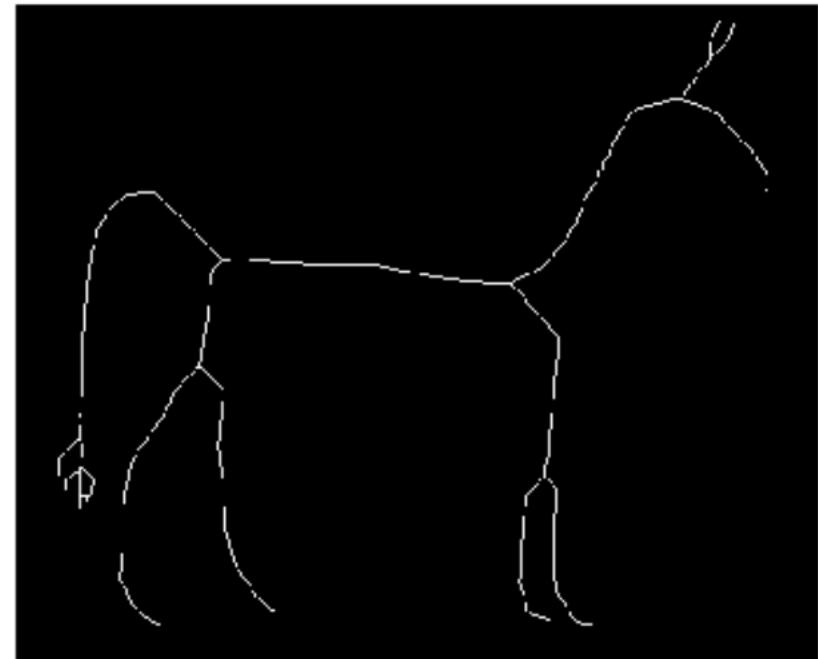
Thinning (Skeletonize)

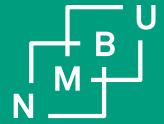


original



skeleton



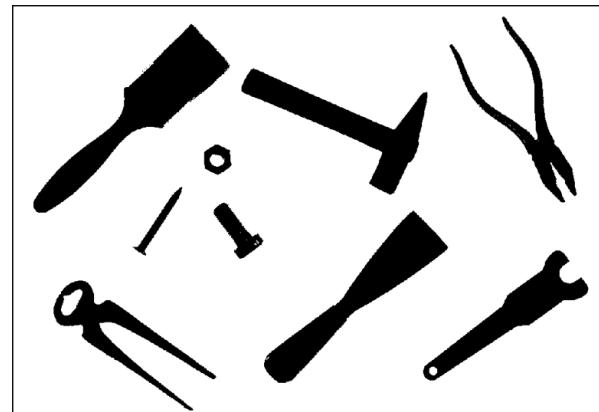


INF250

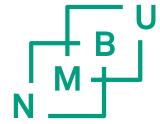
Regions and objects

Foreground, Background

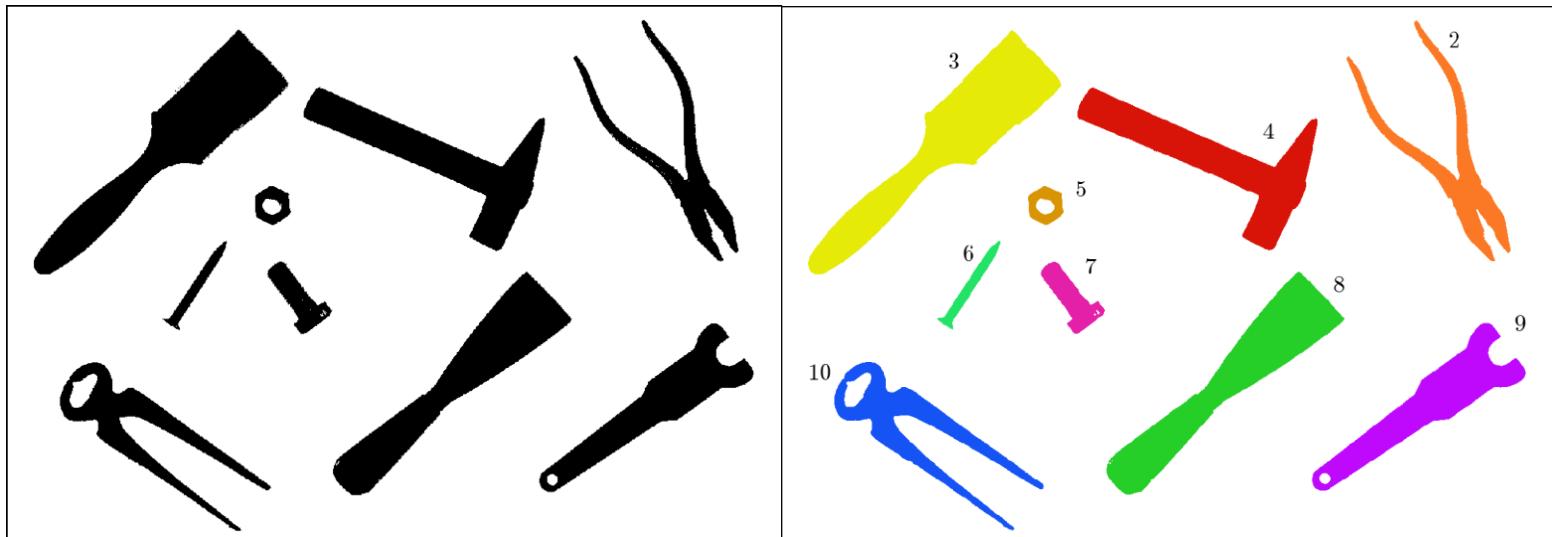
- In a binary image:
 - Pixels with values 1(255) belongs to the foreground and pixels with value 0 belongs to the background
 - Objects are part of neighboring structures or sample
- Our goal is to define objects and their shape
- Objects are defined by pixels that touch each other in coupled binary regions



Contours



1. Find object
2. Define contours of the objects



How to find regions ?



- An important task is to find which pixels defines a region and how many regions there are in the image
- Two methods are described in the book
 - «Flood Filling»
 - «Sequential region marking»

Regionlabeling

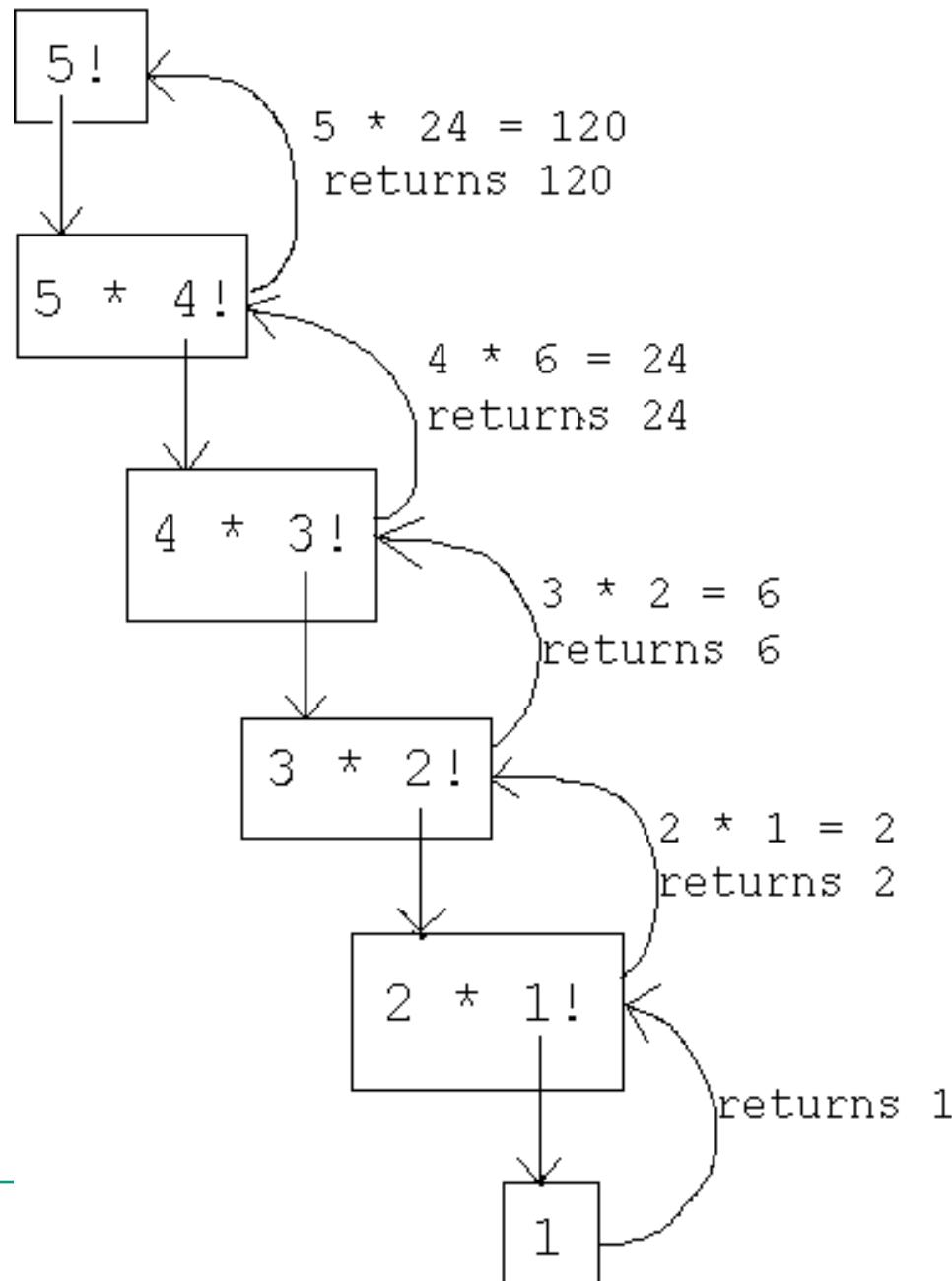
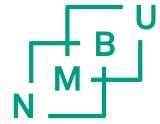


- The underlying algorithm «Flood Fill»:
 - Search for all non marked foreground pixels
 - Fill (mark) the rest of all the neighboring pixels in the region
 - Search for the next pixel outside the earlier found neighborhood
 - Analogy: Water flowing into all the neighboring pixels
 - Different algorithms exist for how the flow is defined
 - Depth-First
 - Breadth-First

Recursive function

- Recursion is programming algorithm where the function calls itself one or several times
- Usually a value is returned from a function call
- A recursive function need a stop criteria (a base) in order to avoid an endless loop
- A common example is the factorial function
 - $N!$
 - $n! = n * (n-1)!$, if $n > 1$ and $f(1) = 1$

Final value = 120



```
1 def factorial(n):  
2     print("factorial has been called with n = " + str(n))  
3     if n == 1:  
4         return 1  
5     else:  
6         res = n * factorial(n-1)  
7         print("intermediate result for ", n, " * factorial(", n-1, "): ", res)  
8         return res  
9  
10 print(factorial(5))
```

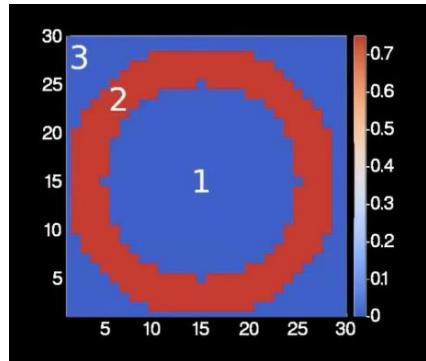
Python

In [5]: %run /Users/kkvaal/CloudStation/ImageryPy/pyAMT/faktor.py

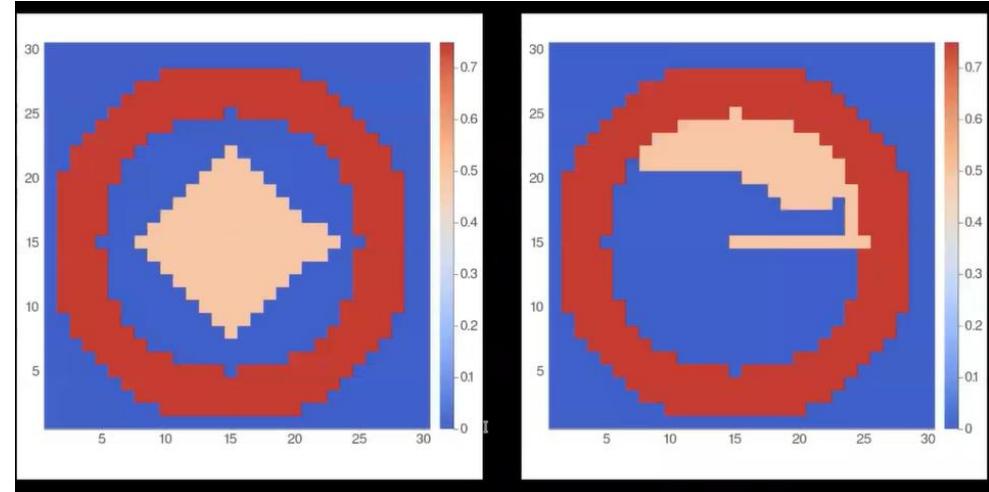
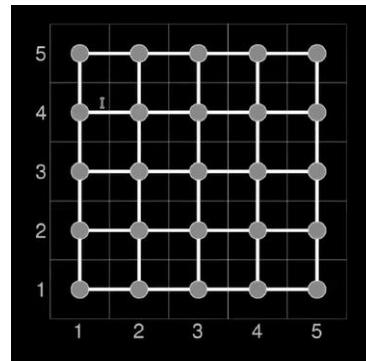
```
factorial has been called with n = 5  
factorial has been called with n = 4  
factorial has been called with n = 3  
factorial has been called with n = 2  
factorial has been called with n = 1  
('intermediate result for ', 2, ' * factorial(', 1, '): ', 2)  
('intermediate result for ', 3, ' * factorial(', 2, '): ', 6)  
('intermediate result for ', 4, ' * factorial(', 3, '): ', 24)  
('intermediate result for ', 5, ' * factorial(', 4, '): ', 120)  
120
```

Flood Fill

Finding domain

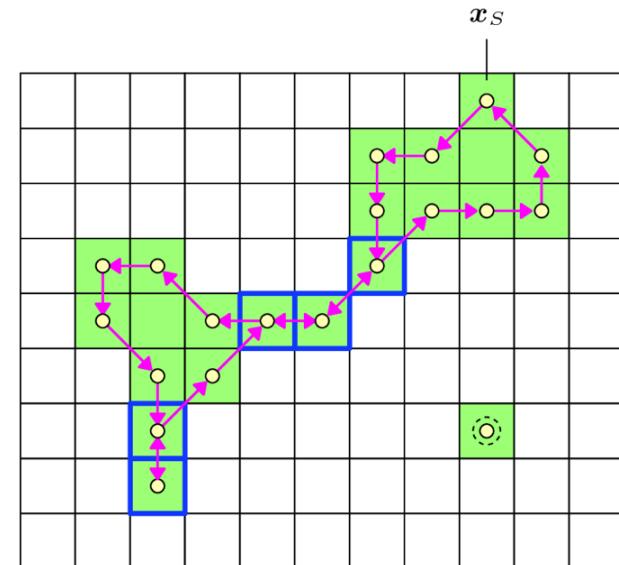
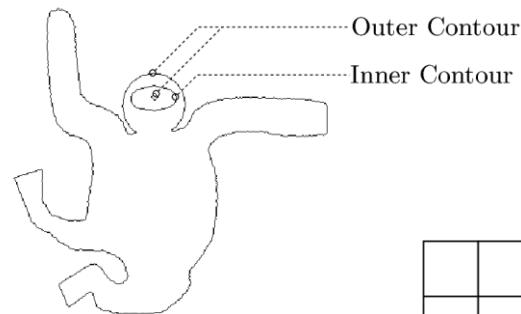


Filling domain

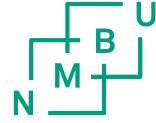


Region contours

- Inner and outer contour



Representation of areas



- Run Length Encoding
- Easy and clear
- Used in data compression

$$Run_i = \langle \text{row}_i, \text{column}_i, \text{length}_i \rangle$$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | |
| 1 | | | × | × | × | × | × | × | × |
| 2 | | | | | | | | | |
| 3 | | | | | × | × | × | × | |
| 4 | | × | × | × | × | | × | × | × |
| 5 | × | × | × | × | × | × | × | × | × |
| 6 | | | | | | | | | |

Bitmap

RLE

$\langle \text{row}, \text{column}, \text{length} \rangle$



$\langle 1, 2, 6 \rangle$

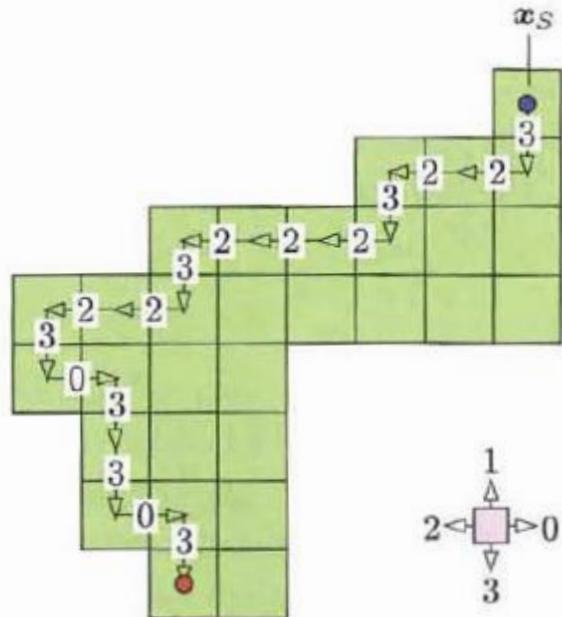
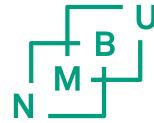
$\langle 3, 4, 4 \rangle$

$\langle 4, 1, 3 \rangle$

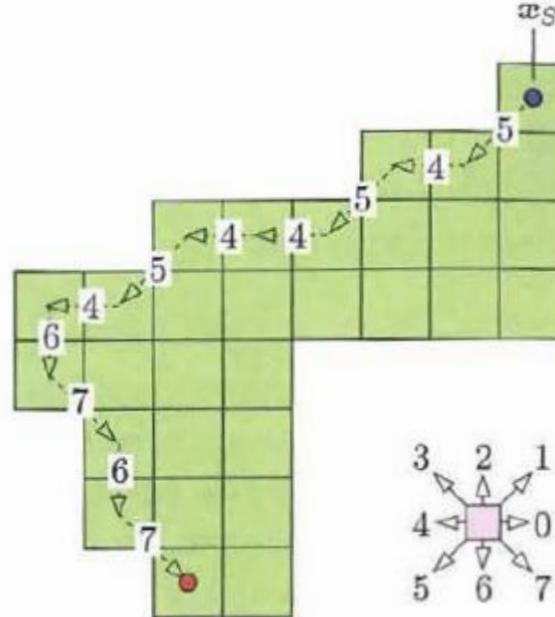
$\langle 4, 5, 3 \rangle$

$\langle 5, 0, 9 \rangle$

Chain Code



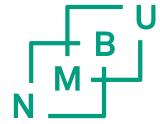
4-Chain Code
 3223222322303303...111
length = 28



8-Chain Code
54544546767...222
 $length = 18 + 5\sqrt{2} \approx 25$

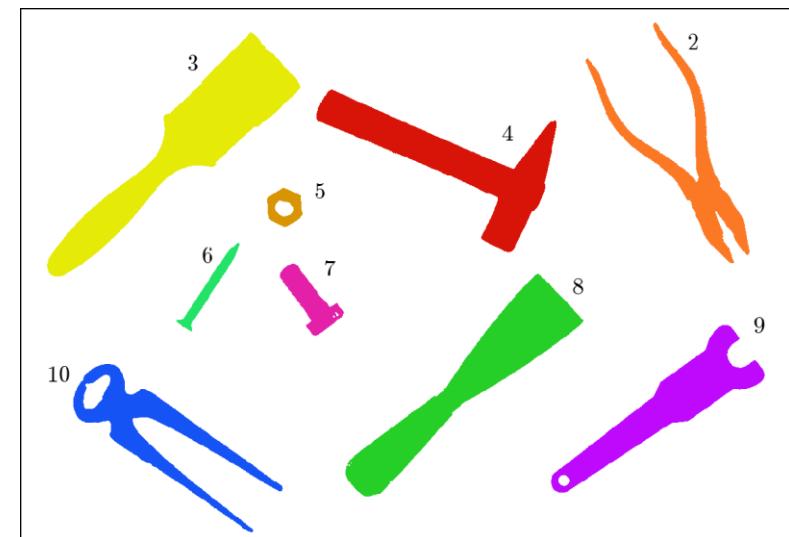
Figure 2.14 Chain codes with 4- and 8-connected neighborhoods. To compute a chain code, begin traversing the contour from a given starting point α_S . Encode the relative position between adjacent contour points using the directional code for either 4-connected (left) or 8-connected (right) neighborhoods. The length of the resulting path, calculated as the sum of the individual segments, can be used to approximate the true length of the contour.

Contours and statistics



- After all regions and contours are found the statistics of the regions can be computed

| Label | Area (pixels) | Bounding Box (left, top, right, bottom) | Center (x_c, y_c) |
|-------|------------------|--|--------------------------|
| 2 | 14978 | (887, 21, 1144, 399) | (1049.7, 242.8) |
| 3 | 36156 | (40, 37, 438, 419) | (261.9, 209.5) |
| 4 | 25904 | (464, 126, 841, 382) | (680.6, 240.6) |
| 5 | 2024 | (387, 281, 442, 341) | (414.2, 310.6) |
| 6 | 2293 | (244, 367, 342, 506) | (294.4, 439.0) |
| 7 | 4394 | (406, 400, 507, 512) | (454.1, 457.3) |
| 8 | 29777 | (510, 416, 883, 765) | (704.9, 583.9) |
| 9 | 20724 | (833, 497, 1168, 759) | (1016.0, 624.1) |
| 10 | 16566 | (82, 558, 411, 821) | (208.7, 661.6) |



Bounding box og Convex Hull

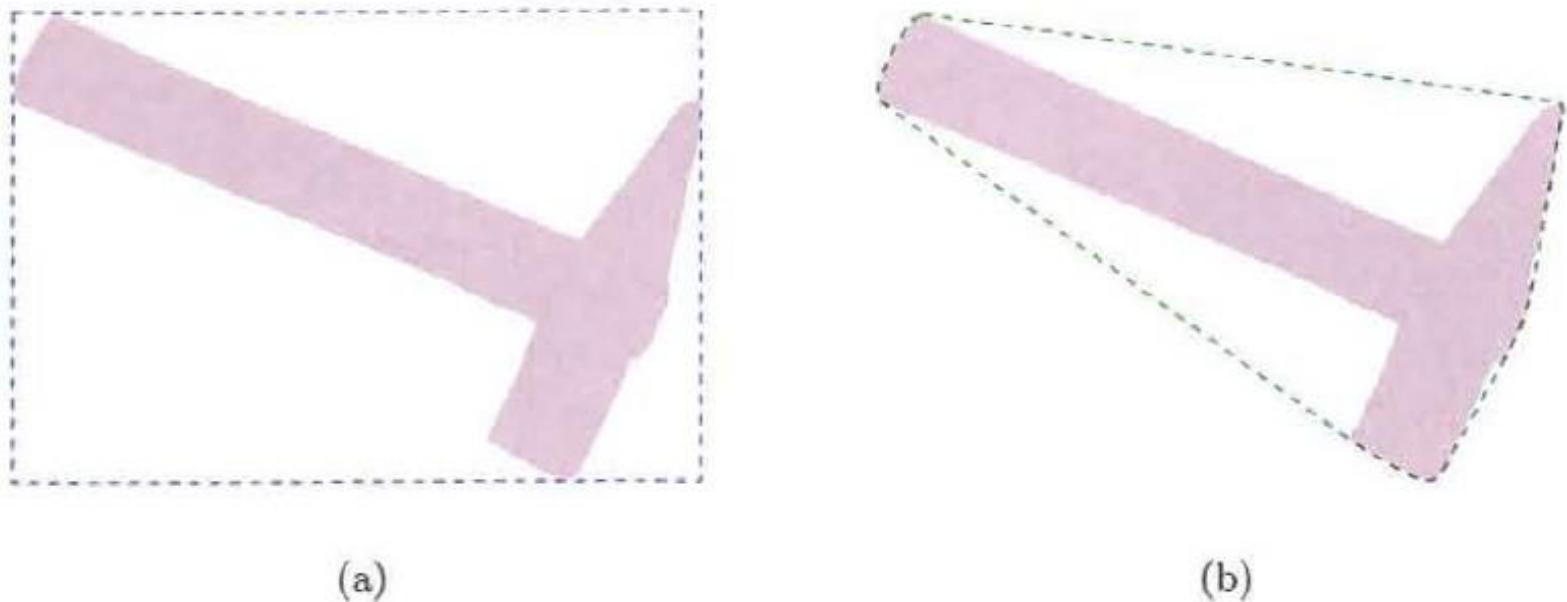


Figure 2.16 Example bounding box (a) and convex hull (b) of a binary image region.



Geometrical features

- Perimeter: $P(R) = \sum_{i=0}^{M-1} \text{length}(C_i)$



= 1 when $c=0,2,4,6$

= $\sqrt{2}$ when $c=1,3,5,7$

- Area: $A(R) = N$ (number of pixels)

- Circularity $C(R) = 4\rho \times \frac{A(R)}{P^2(R)}$

Circularity(sirkularitet)

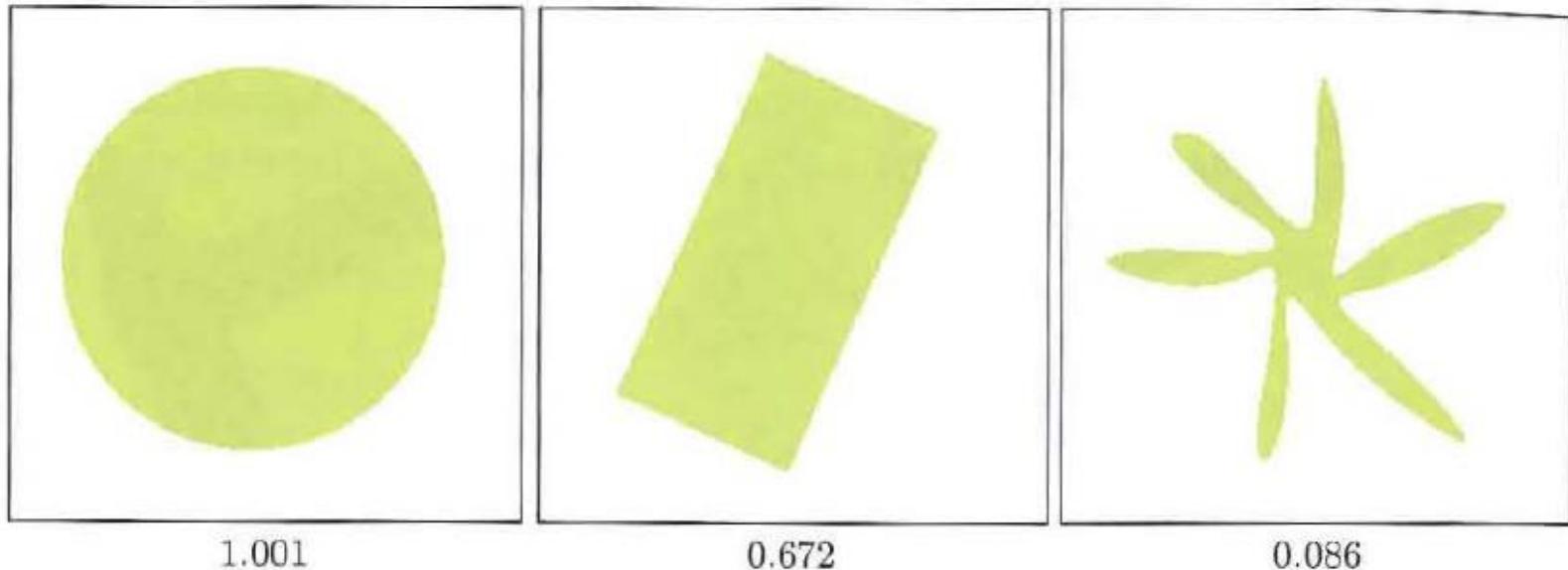
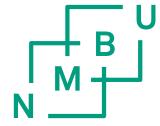
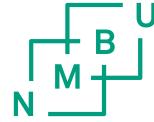


Figure 2.15 Circularity values for different shapes. Shown are the corresponding estimates for $\text{Circularity}(\mathcal{R})$ as defined in Eqn. (2.12).



Circularity vs Roundness

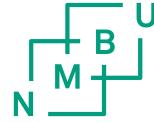
There is a difference between the terms «Circularity» and «Roundness» (not described in the book)

«Circularity» is computed from the perimeter whereas
«Roundness» is computed from the major axis

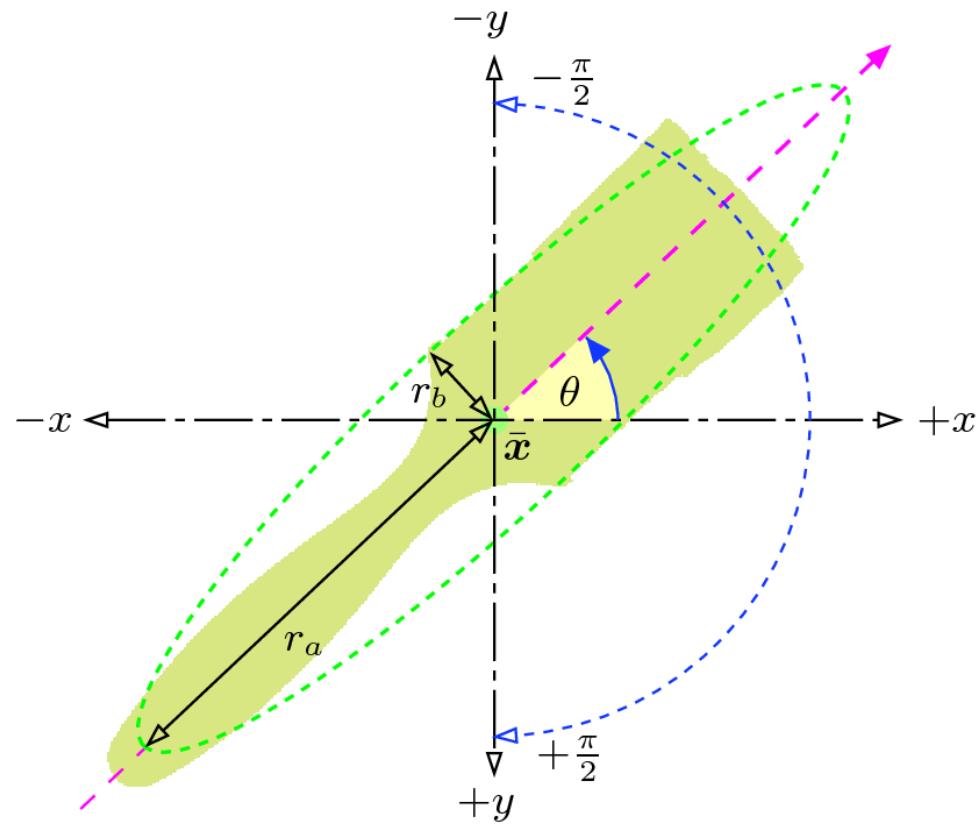
$$\text{Circularity} = 4 \times \frac{\text{Area}}{\text{Perimeter}^2}$$

$$\text{Roundness} = 4 \times \frac{\text{Area}}{\rho * (\text{Major_axis})^2}$$

Other statistical shape descriptors



- Centroid
- Moments
- Central moments
- Directions
-



Projections

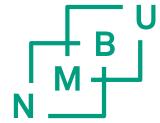


Figure 2.20 Example of the horizontal projection $P_{\text{hor}}(v)$ (right) and vertical projection $P_{\text{ver}}(u)$ (bottom) of a binary image.

Image moments

Moments summarize a shape given image $I(x, y)$:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Central moments are translation invariant:

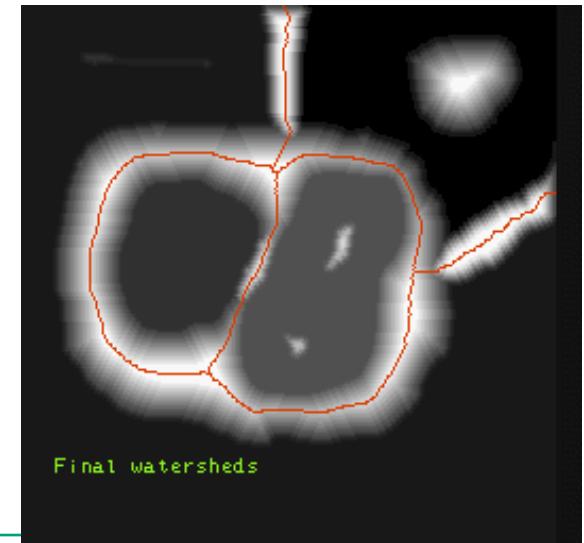
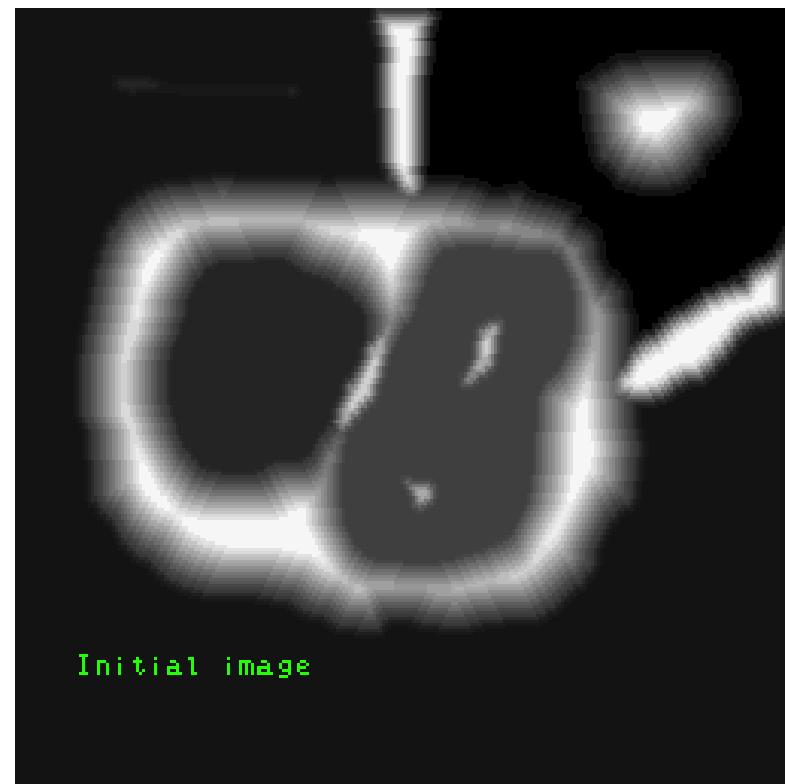
$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

$$\bar{x} = \frac{M_{10}}{M_{00}} \quad \bar{y} = \frac{M_{01}}{M_{00}}$$

https://www.youtube.com/watch?v=AAbUfZD_09s

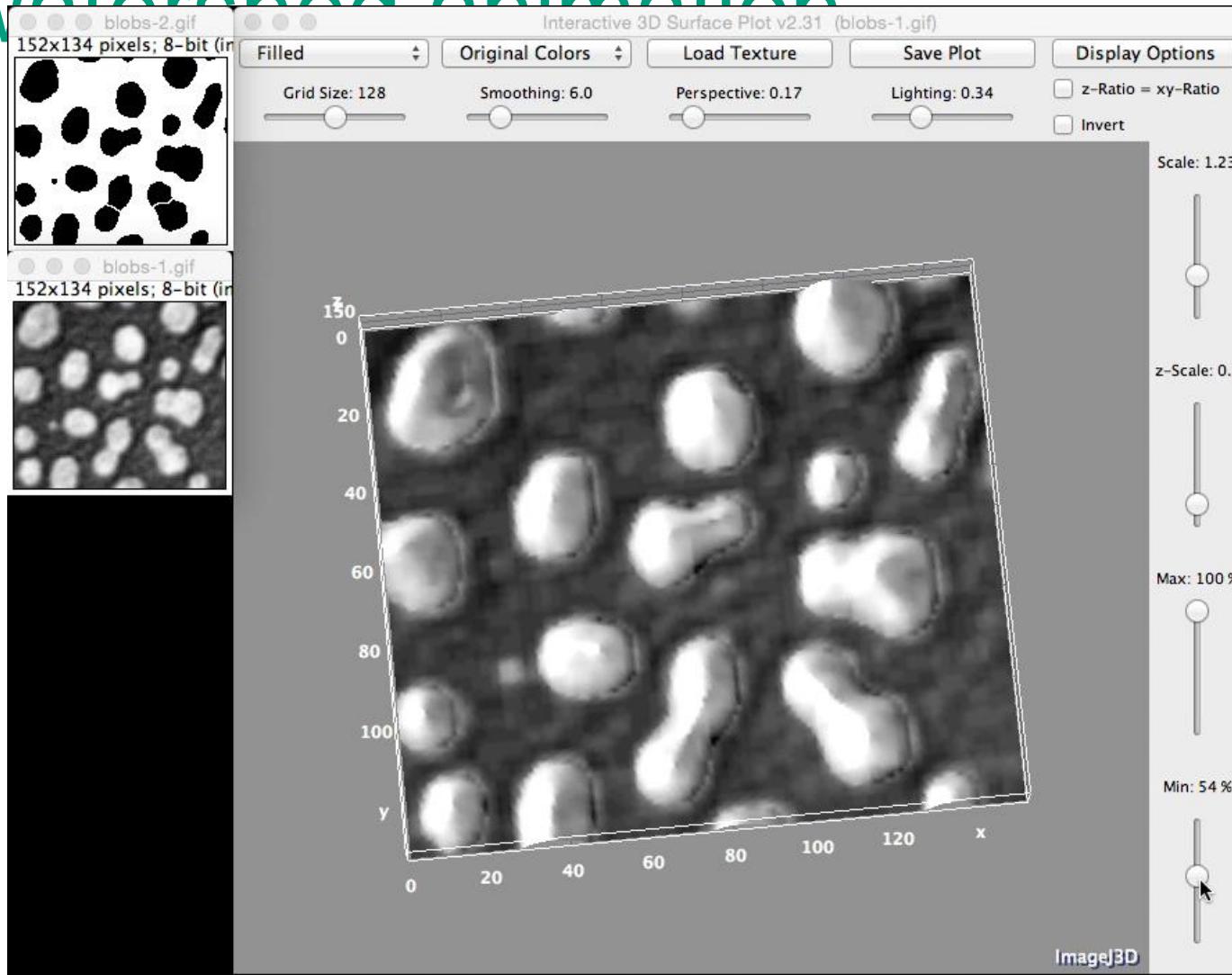
Watershed - segmentation

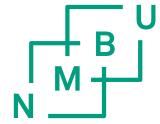
- Grey level images can be viewed as a topologic map
- Binary images can be transformed to a topologic map by using the distance from the perimeter to the center as a weight
- Morphologic operations
 - Fill water from a minimum point and build a shed so that the water doesn't flood inn from other areas
- This gives us two dataset:
 - Water basins
 - Floodlines



Watershed animation

U
B
M
N



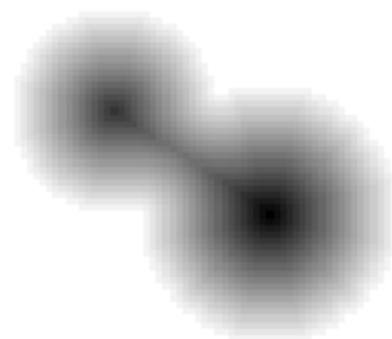


Watershed - Python

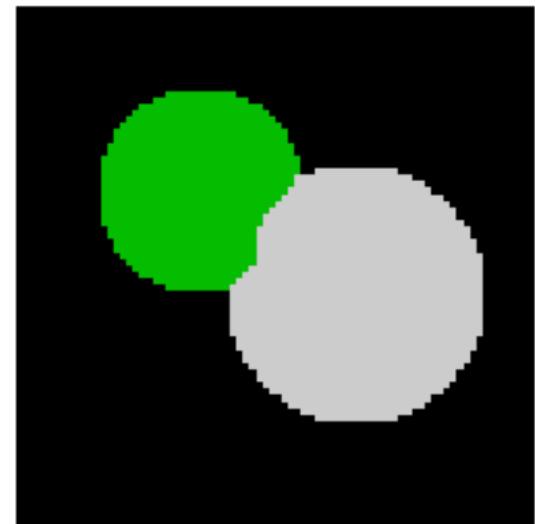
Overlapping objects



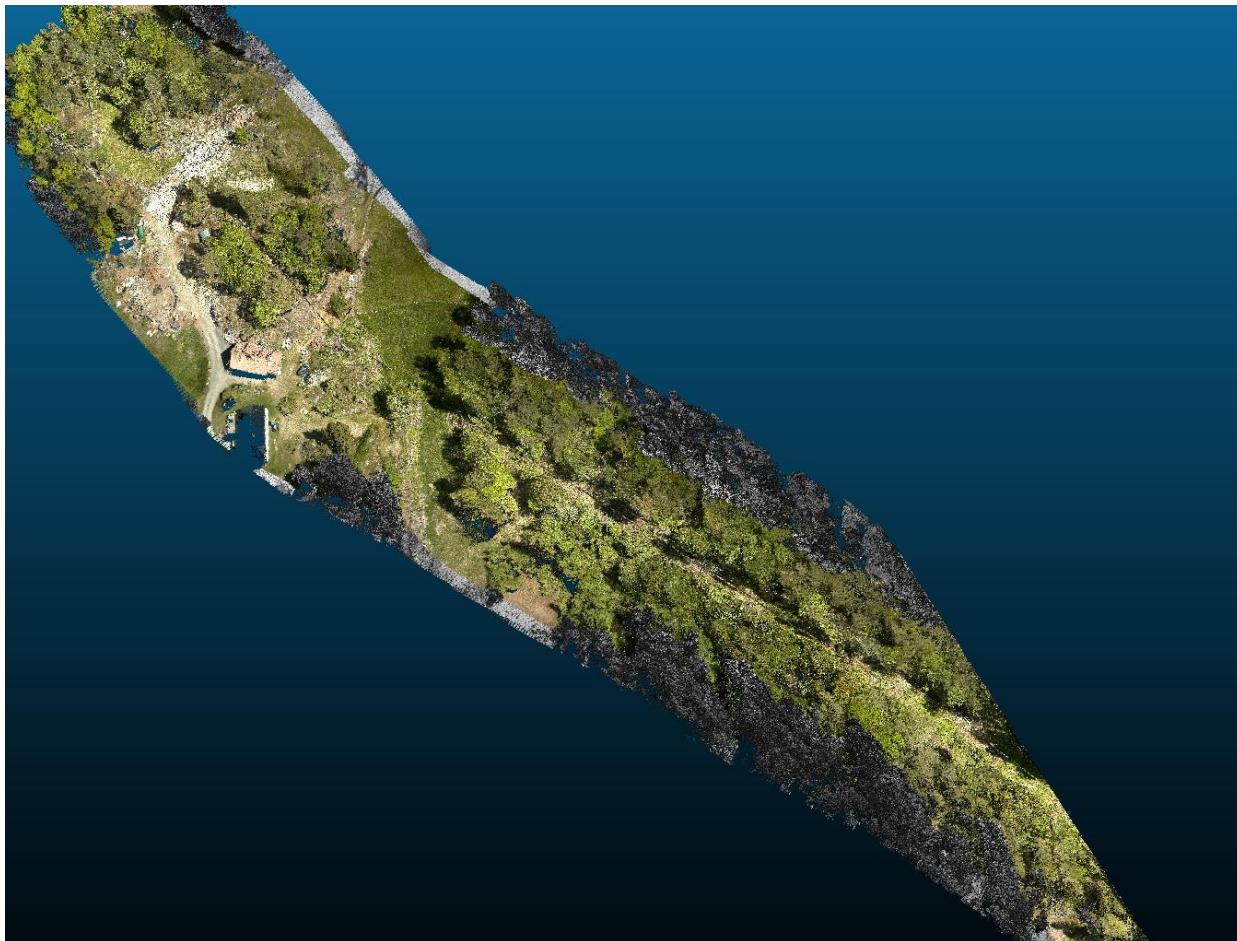
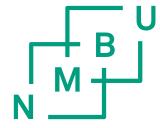
Distances

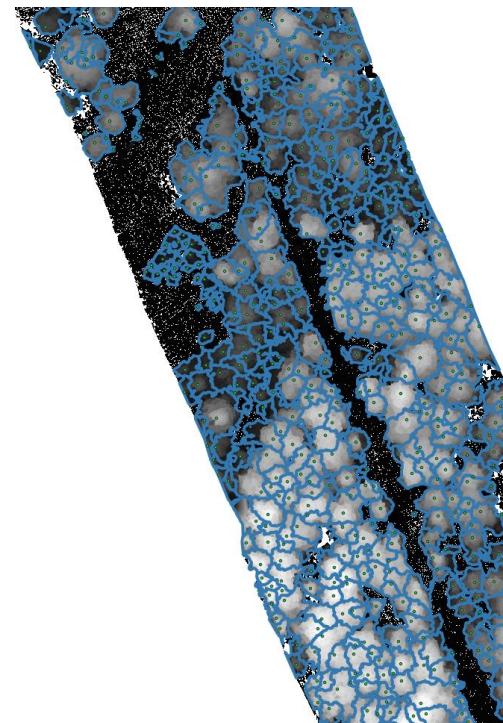
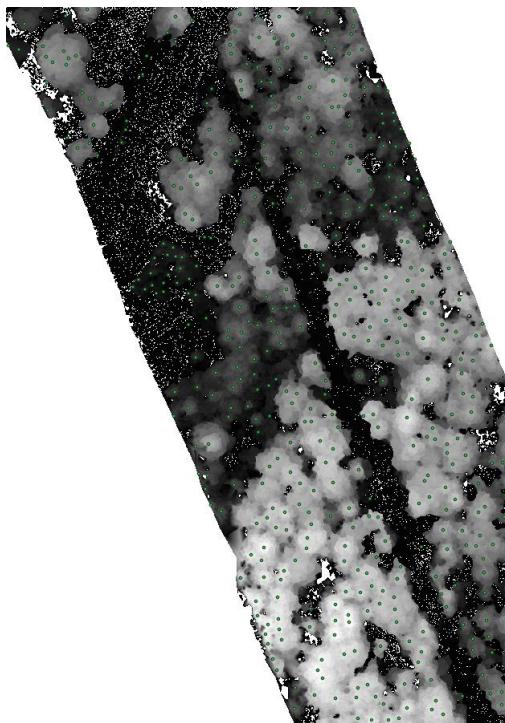
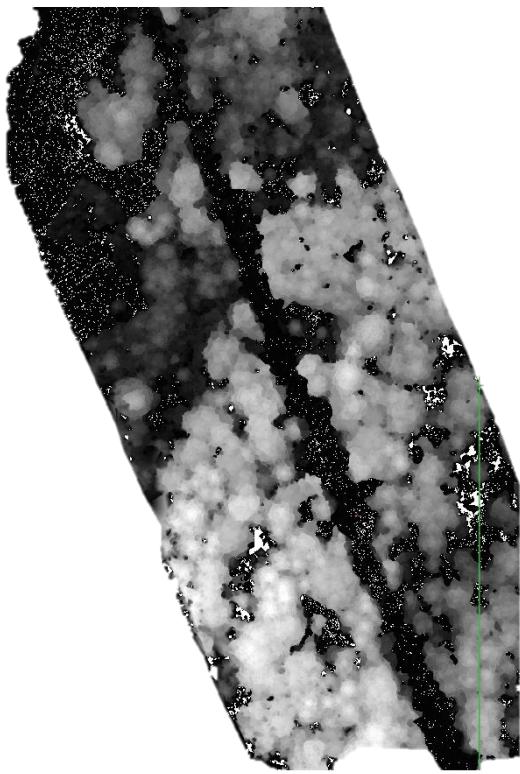


Separated objects



http://www.scipy-lectures.org/advanced/image_processing/auto_examples/plot_watershed_segmentation.html





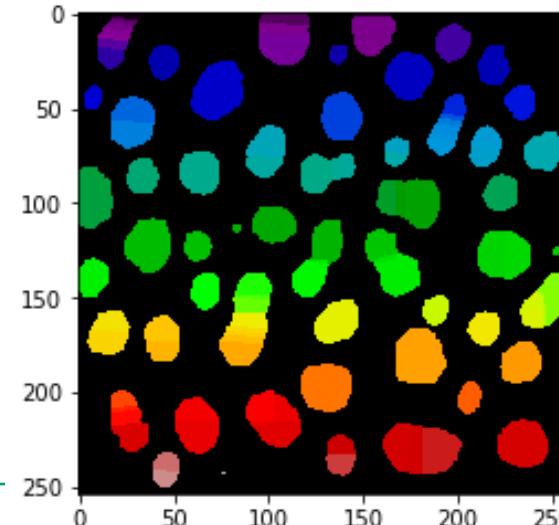
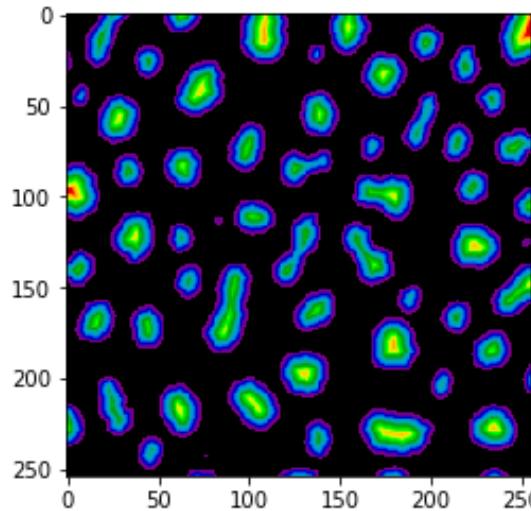
```
from skimage.morphology import watershed
import matplotlib.pyplot as plt
import numpy as np
from skimage.feature import peak_local_max
from scipy import ndimage as ndi
from skimage import io

filename = 'blobs.tif'

blob = io.imread(filename)
plt.imshow(blob, 'gray')

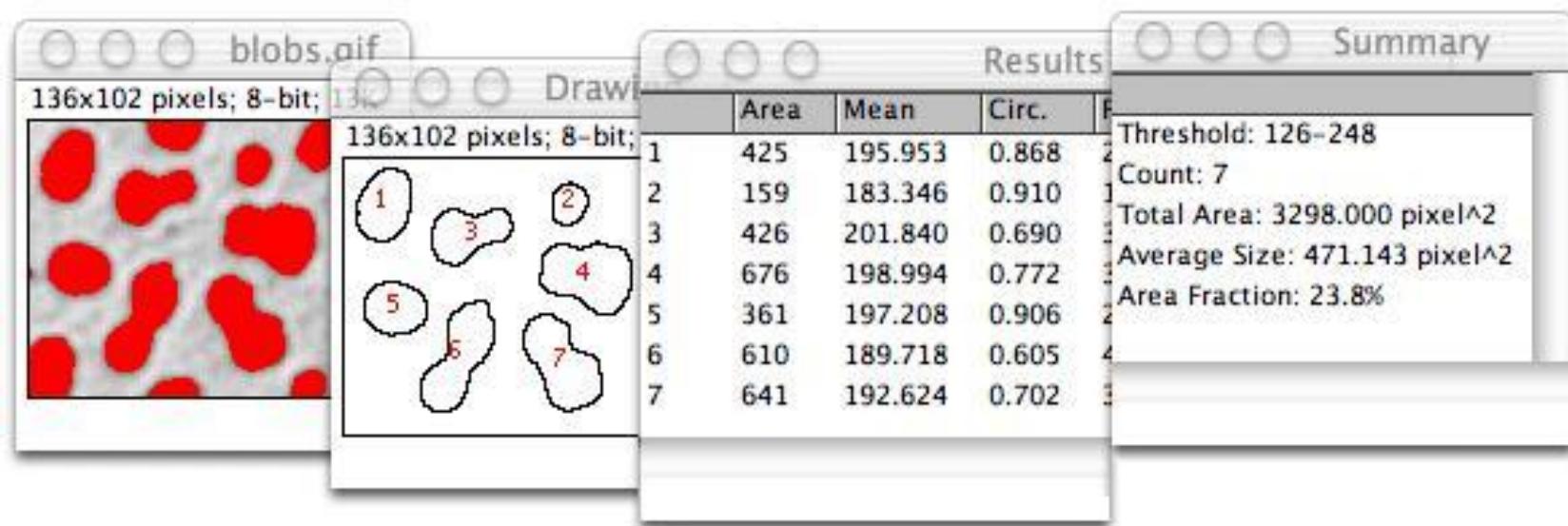
distance = ndi.distance_transform_edt(blob)
local_maxi = peak_local_max(distance, indices=False, footprint=np.ones((3, 3)),
                             labels=blob)
markers = ndi.label(local_maxi)[0]
labels = watershed(-distance, markers, mask=blob)

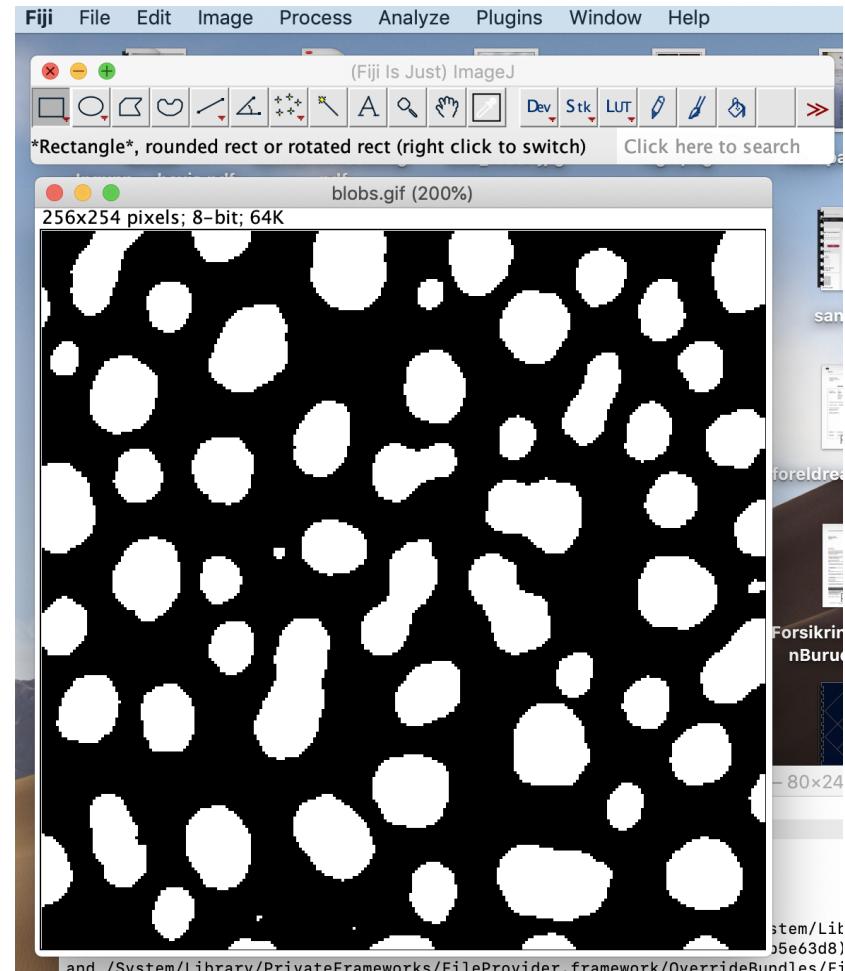
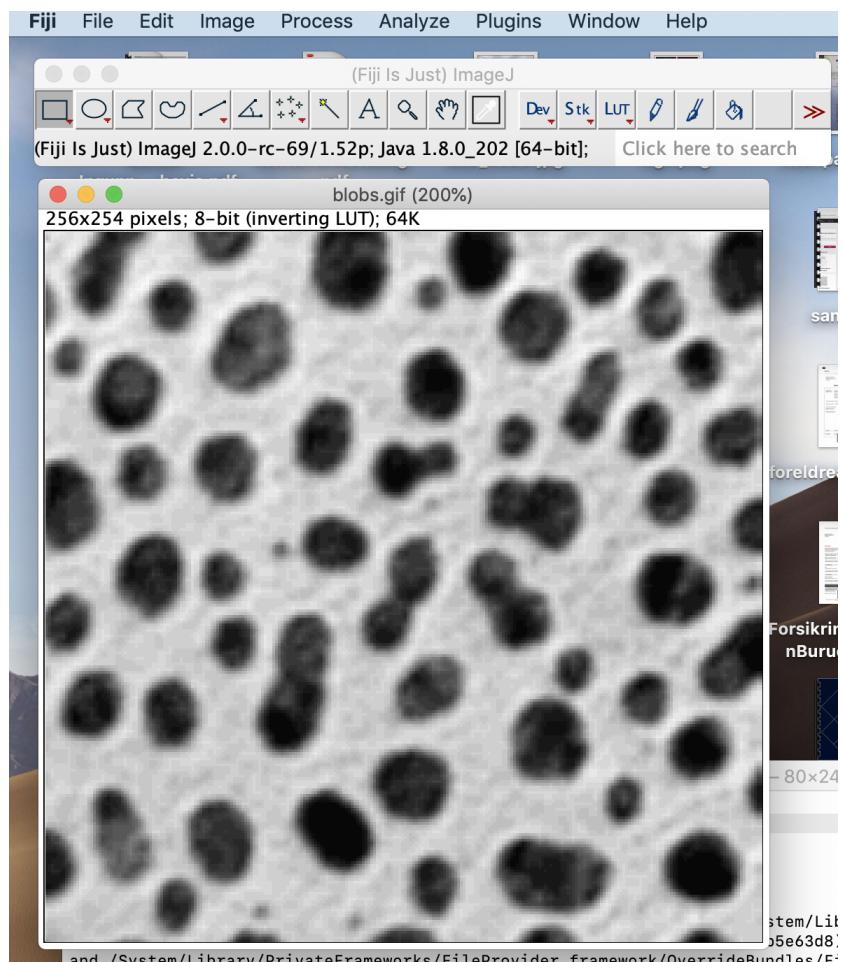
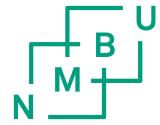
plt.imshow(labels, 'spectral')
```

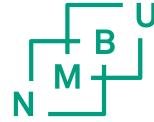
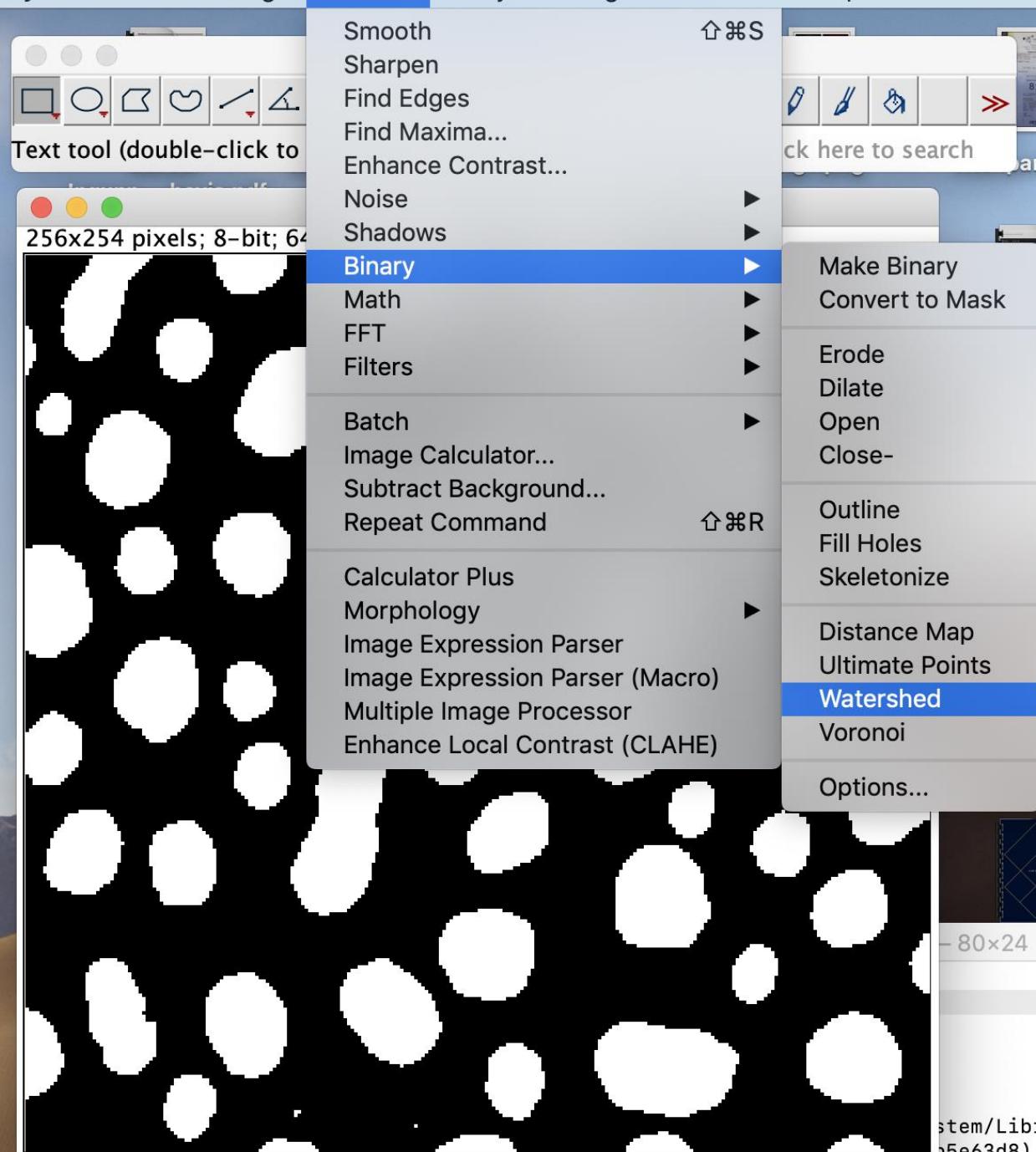


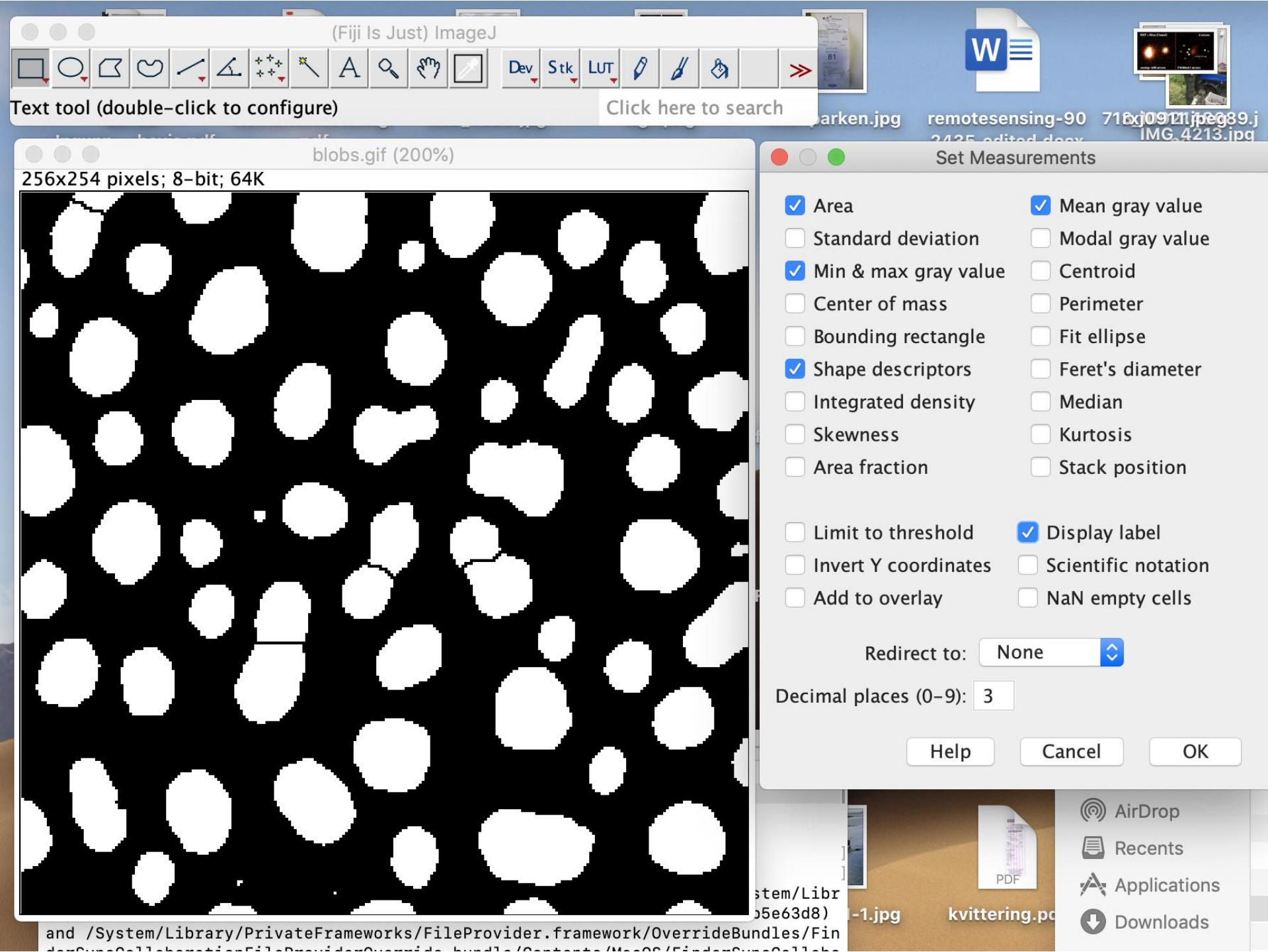
Analyzis of particles

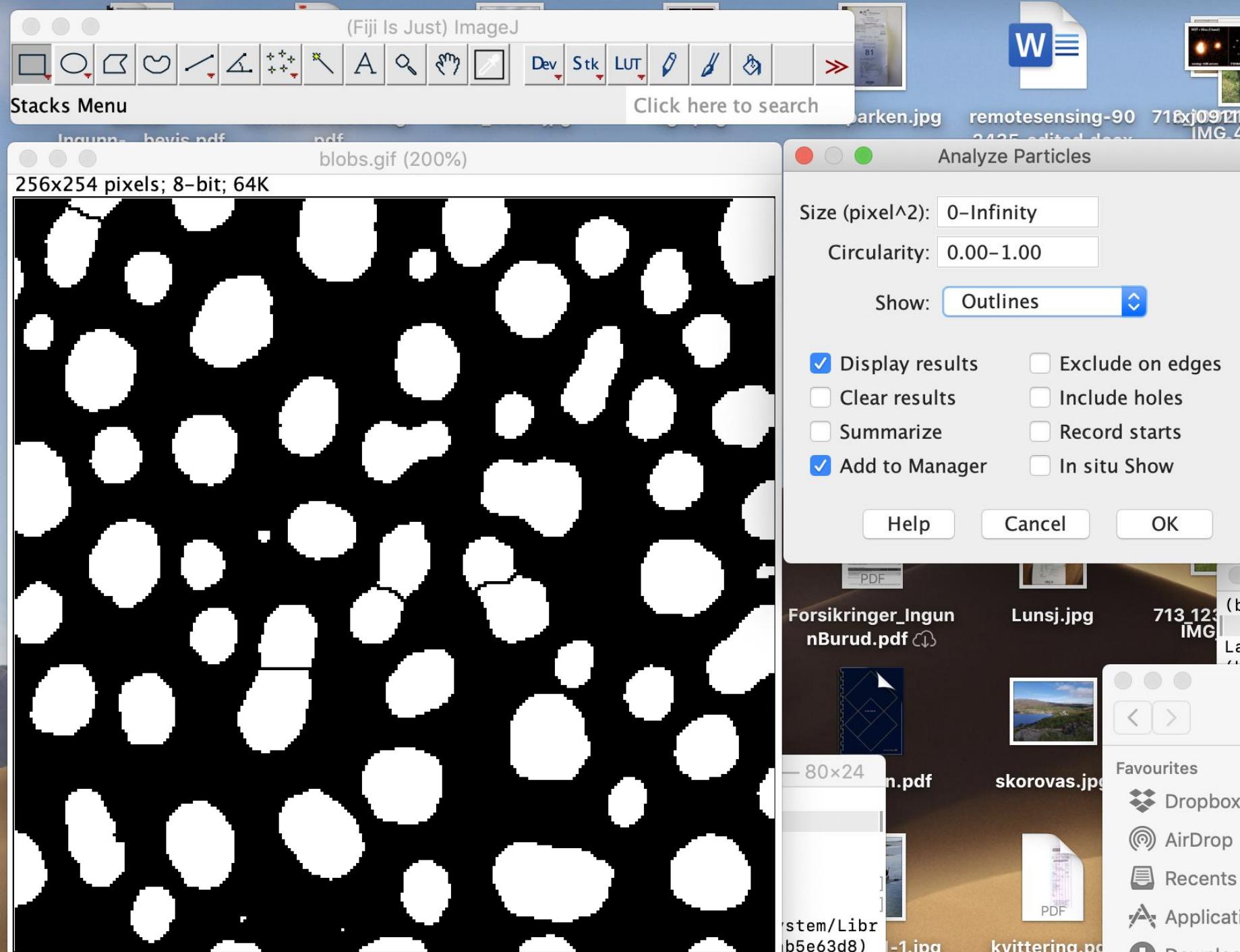
http://imagejdocu.tudor.lu/doku.php?id=gui:analyze:analyze_particles













Dev Stk LUT



Click here to search

Burud-

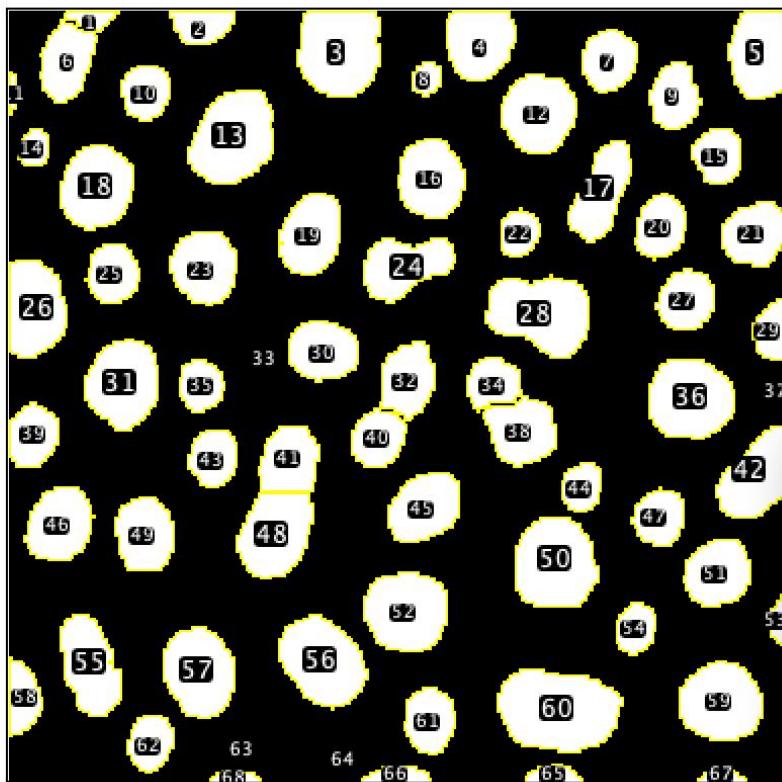
Kontantutbetaling.

IMG 2590.jpg

file

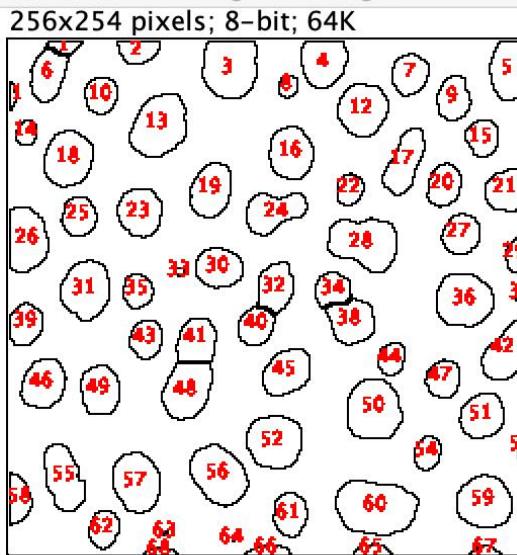
Results

blobs.gif (150%)
256x254 pixels; 8-bit; 64K



| | Area | Mean | Min | Max | Circ. | AR | Round | Solidity |
|----|------|------|-----|-----|-------|-------|-------|----------|
| 1 | 76 | 255 | 255 | 255 | 0.562 | 2.522 | 0.396 | 0.884 |
| 2 | 185 | 255 | 255 | 255 | 0.763 | 1.778 | 0.562 | 0.951 |
| 3 | 658 | 255 | 255 | 255 | 0.872 | 1.068 | 0.936 | 0.967 |
| 4 | 434 | 255 | 255 | 255 | 0.884 | 1.064 | 0.940 | 0.959 |
| 5 | 477 | 255 | 255 | 255 | 0.821 | 1.570 | 0.637 | 0.968 |
| 6 | 341 | 255 | 255 | 255 | 0.757 | 1.603 | 0.624 | 0.905 |
| 7 | 285 | 255 | 255 | 255 | 0.926 | 1.153 | 0.867 | 0.938 |
| 8 | 81 | 255 | 255 | 255 | 0.971 | 1.204 | 0.830 | 0.926 |
| 9 | 278 | 255 | 255 | 255 | 0.854 | 1.389 | 0.720 | 0.919 |
| 10 | 231 | 255 | 255 | 255 | 0.936 | 1.141 | 0.877 | 0.941 |
| 11 | 30 | 255 | 255 | 255 | 0.406 | 4.338 | 0.230 | 0.870 |

Drawing of blobs.gif



ROI Manager

| | |
|-----------|--|
| 0001-0003 | Add [t] |
| 0002-0005 | Update |
| 0003-0014 | Delete |
| 0004-0011 | Rename... |
| 0005-0014 | Measure |
| 0006-0016 | Deselect |
| 0007-0016 | Properties... |
| 0008-0022 | Flatten [F] |
| 0009-0028 | More » |
| 0010-0027 | <input checked="" type="checkbox"/> Show All |
| 0011-0027 | <input checked="" type="checkbox"/> Labels |
| 0012-0034 | |
| 0013-0041 | |
| 0014-0045 | |
| 0015-0048 | |
| 0016-0055 | |
| 0017-0059 | |
| 0018-0058 | |
| 0019-0073 | |

timeliste_signed.p

df

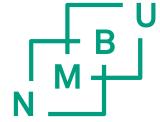


Ingr



iCloud Drive

Record

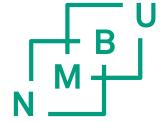


Python - measure

```
from skimage import measure
properties = measure.regionprops(labels)
for prop in properties:
    print(prop.perimeter, prop.area, prop.centroid, prop.eccentricity)
```

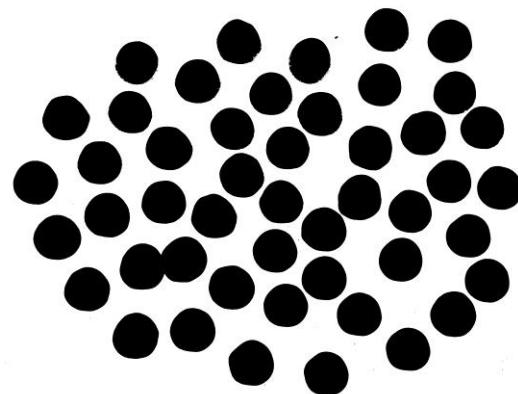
<http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops>

Watershed - Imagej



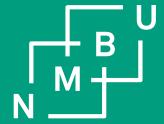
- Watershed on gray level images
 - <http://bigwww.epfl.ch/sage/soft/watershed/>

Masks



Making a binary
(holes=0, background=1)

Masking original image by
combining original OR mask



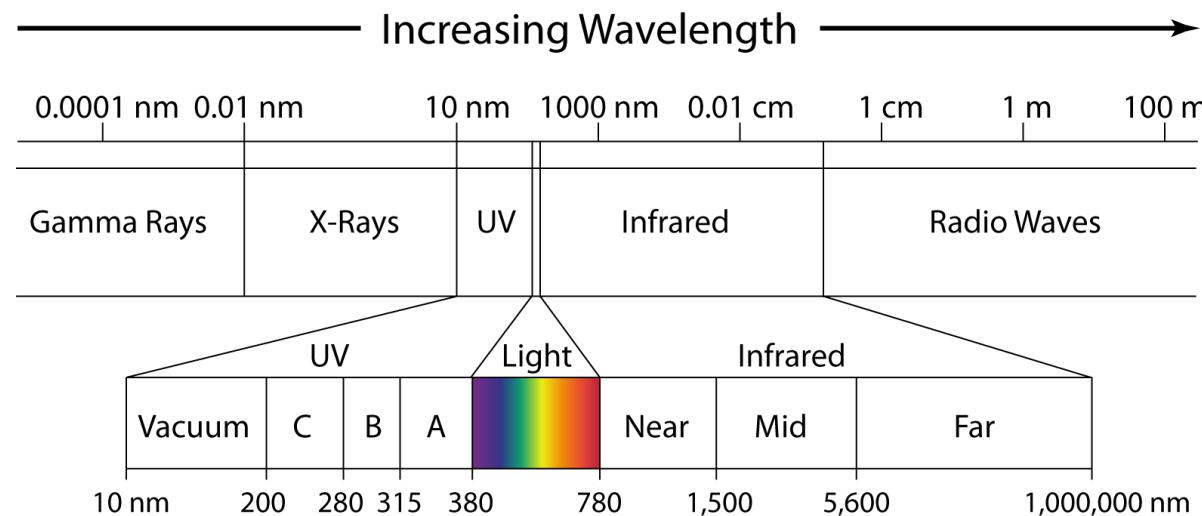
INF250

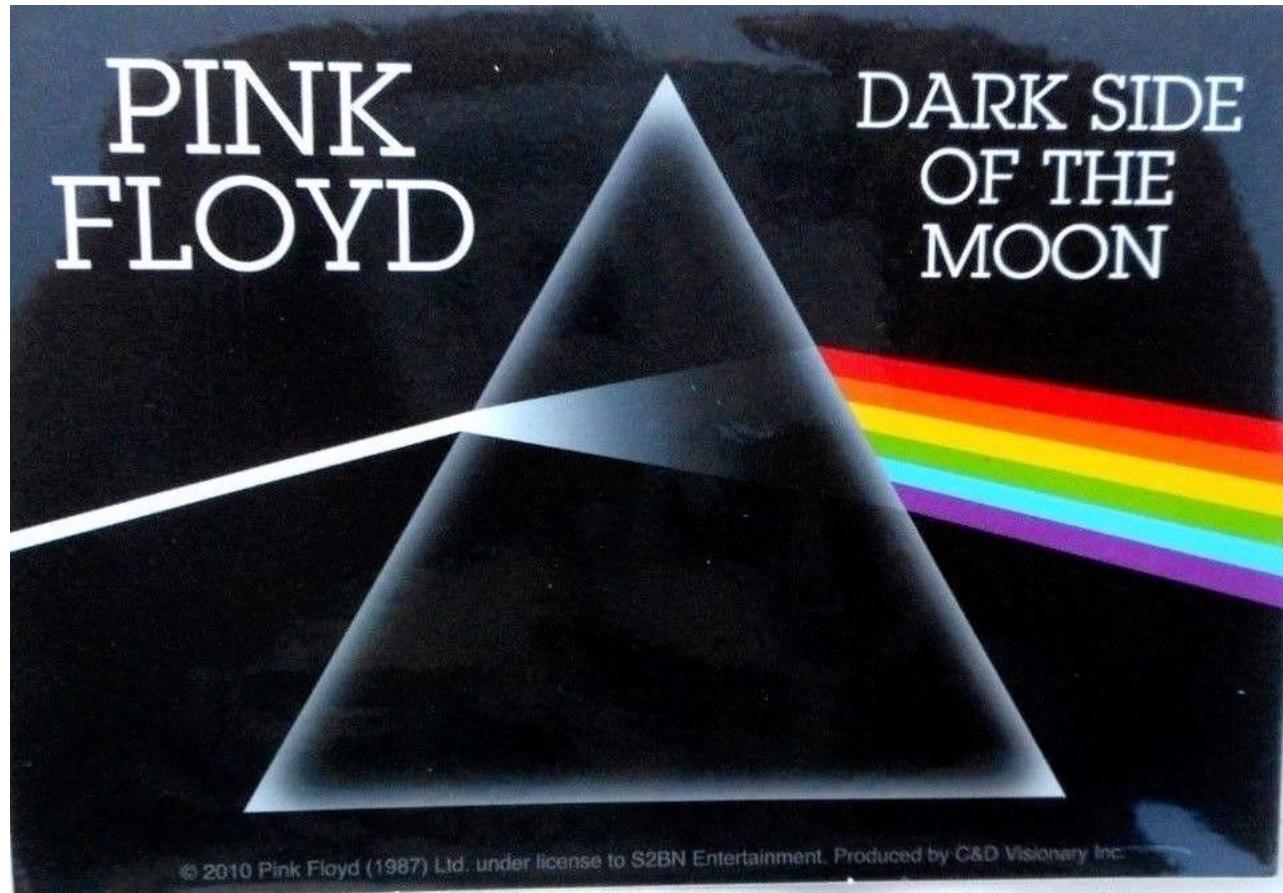
Colour images

Things you need to know

- RGB colour space – what is it and how do we represent it ?
- What is CIE and the CIE horseshoe ?
- Colour models you must be able to describe:
 - CIE Lab colour space
 - RGB
 - Munsell
- Terms to know:
 - Hue
 - Chroma
 - Brightness
 - Lightness
 - Gamut

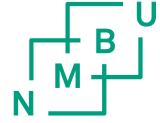
Electromagnetic spectrum





Sir Isaac Newton discovered white light spectrum in 1666,
at the age of 24

RGB colour space



red

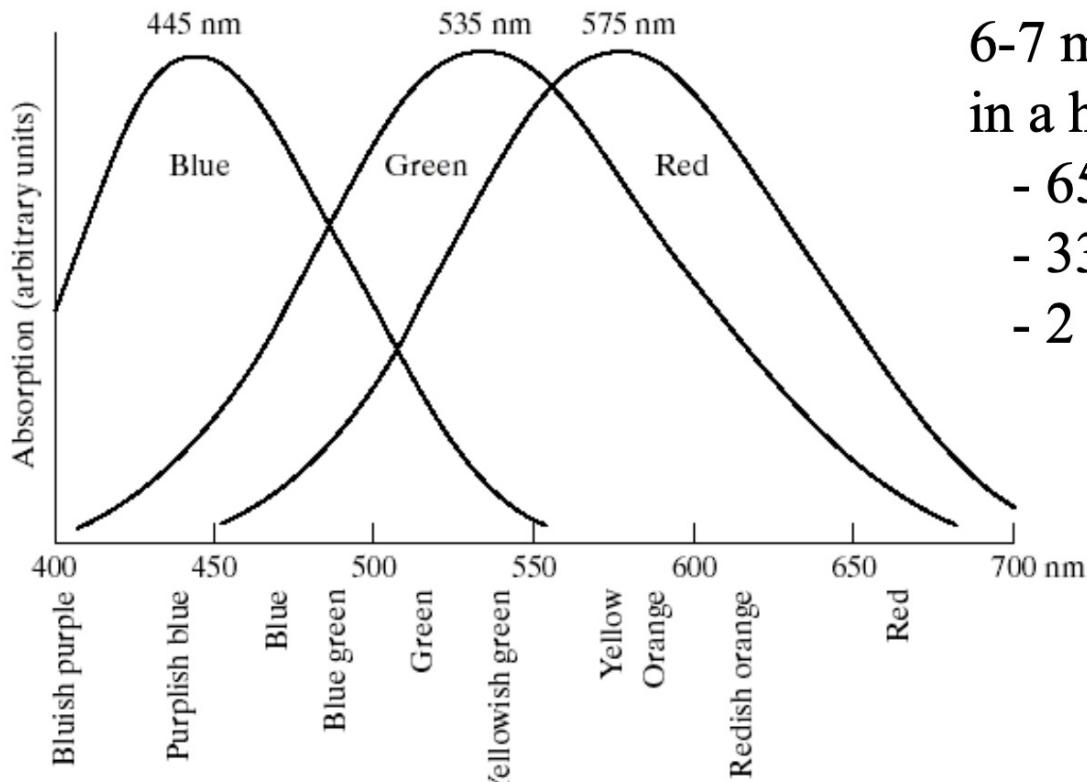


green



blue





6-7 millions cones
in a human eye

- 65% sensitive to **Red light**
- 33% sensitive to **Green light**
- 2 % sensitive to **Blue light**

Primary colors:
Defined CIE in 1931

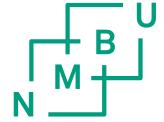
Red = 700 nm

Green = 546.1 nm

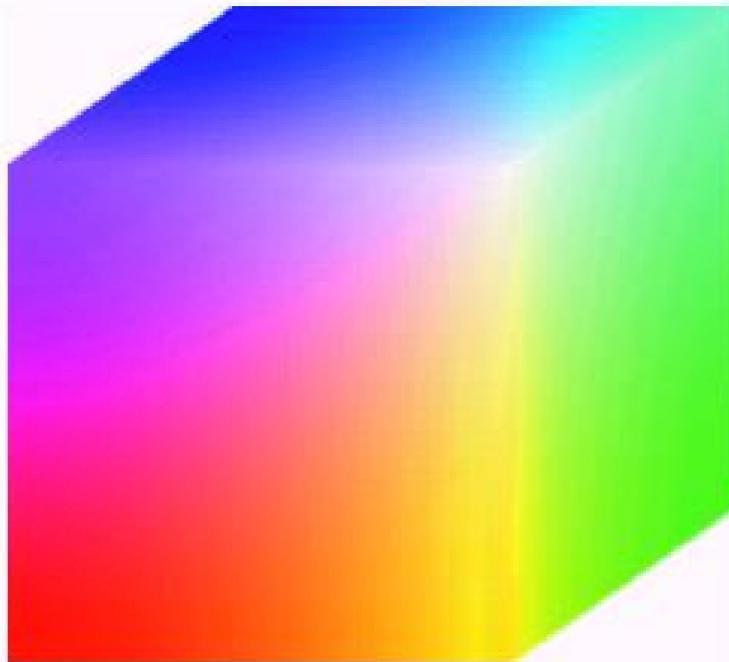
Blue = 435.8 nm

CIE = Commission Internationale de l'Eclairage
(The International Commission on Illumination)

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.)



RGB colour space



R = 8 bits
G = 8 bits
B = 8 bits

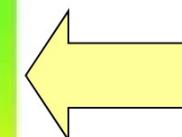
Color depth 24 bits
= 16777216 colors



(R = 0)

(G = 0)

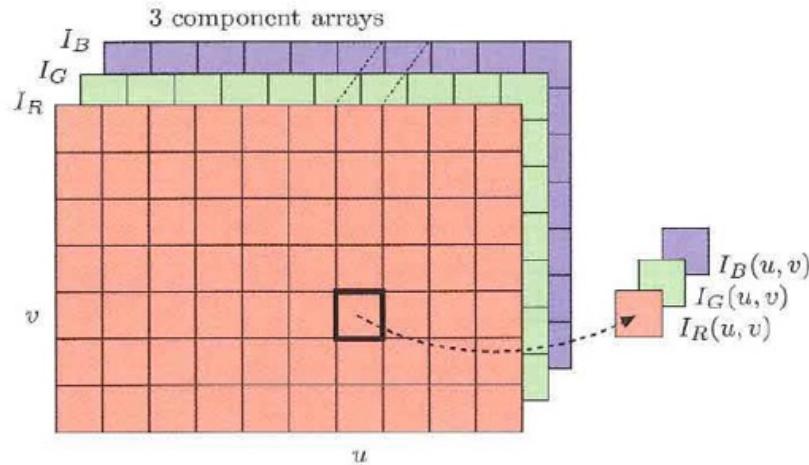
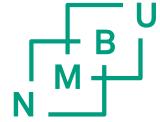
(B = 0)



Hidden faces
of the cube

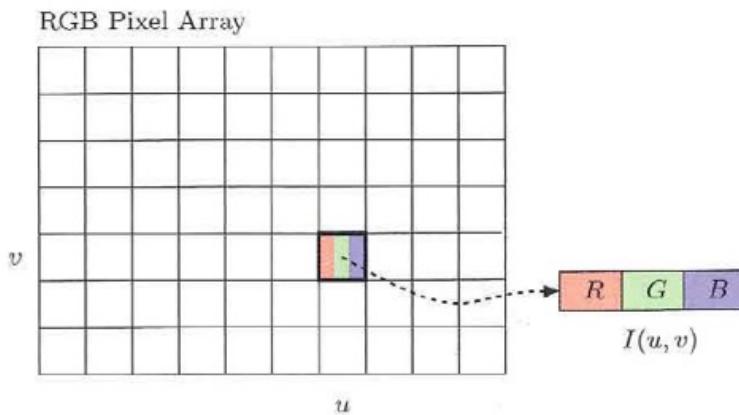
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.)

Two ways of storing colours



Component ordering

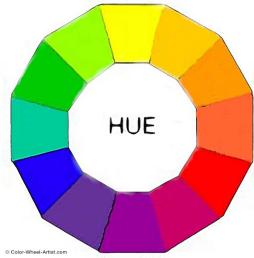
Figure 8.3 RGB color image in component ordering. The three color components are laid out in separate arrays I_R , I_G , I_B of the same size.



Packed ordering

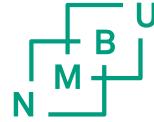
Figure 8.4 RGB-color image using packed ordering. The three color components R , G , and B are placed together in a single array element.

- **Hue** – attribute of colours that permits them to be classed as red, yellow, green, blue or an intermediate between any contiguous pair of these



- **Brightness** – an attribute of visual perception in which an object appears to radiate or reflect light
- **Lightness** – value or tone of a colour
- **Chroma** – a measure of colour purity on the Munsell colour system
- **Gamut** – a complete subset of colours, within a certain colour space or within an output device

http://changingminds.org/explanations/perception/visual/lightness_variants.htm

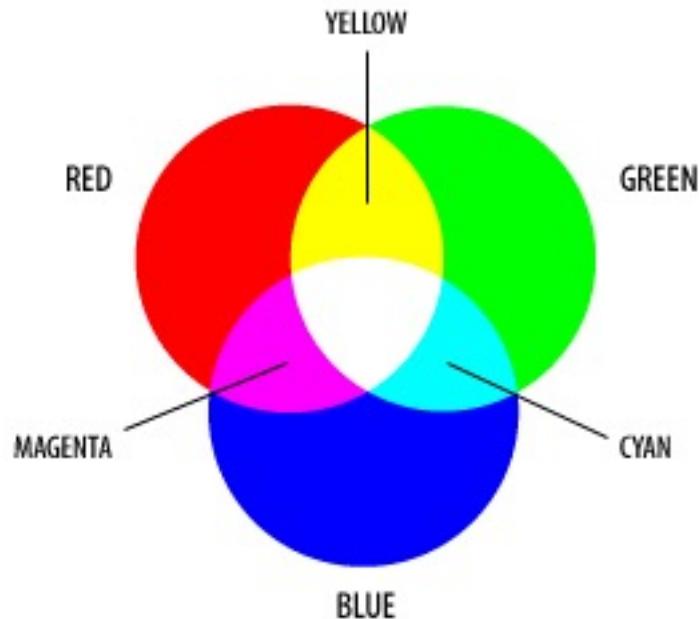


Colour models

- Colour models are used to classify colours and to qualify them according to attributes such as **hue, saturation, chroma, lightness, or brightness**.
- They are further used for matching colors and are valuable resources for anyone working with color in any medium: print, video, or Web.

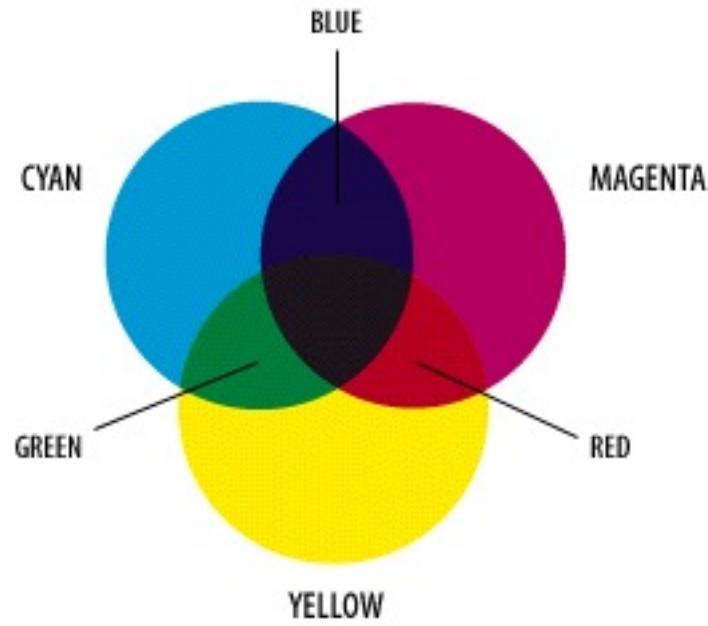
RGB colour model

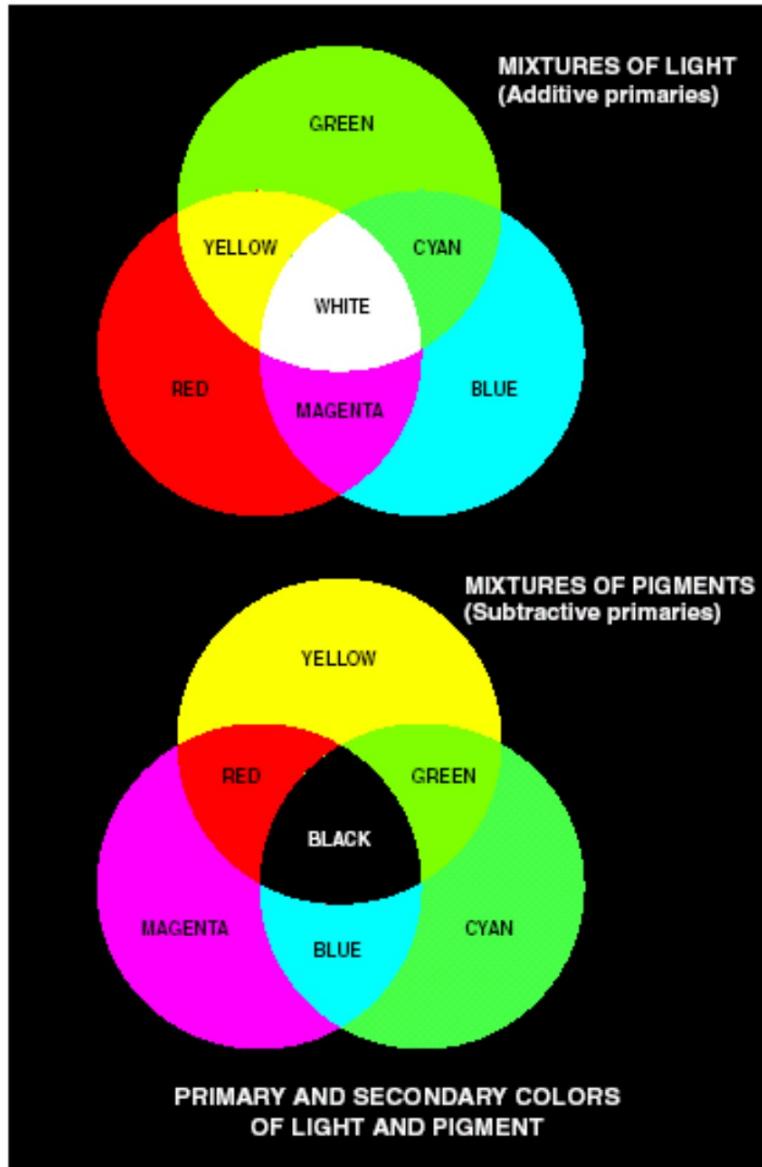
- Additive colours



RGB colour model

- Subtractive colours CMY





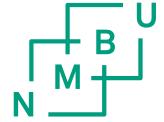
Additive primary colors: RGB
use in the case of light sources
such as color monitors

RGB add together to get white

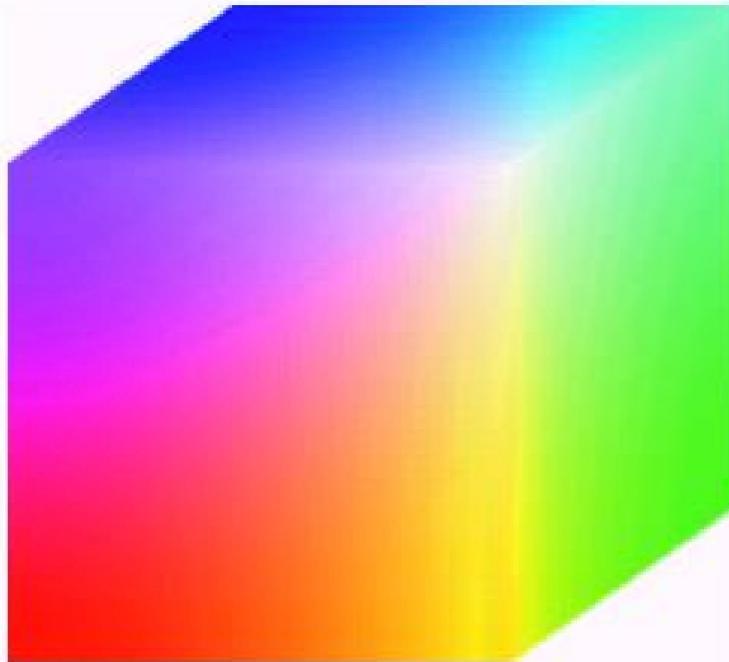
Subtractive primary colors: CMY
use in the case of pigments in
printing devices

White subtracted by CMY to get
Black

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.)

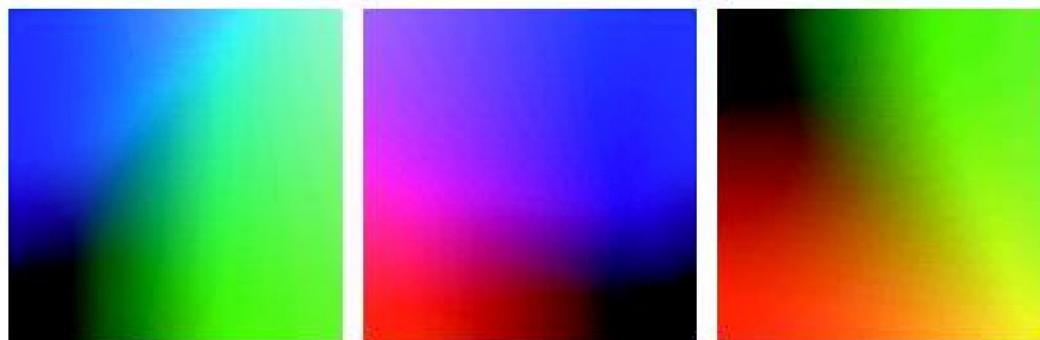


RGB colour space



$R = 8 \text{ bits}$
 $G = 8 \text{ bits}$
 $B = 8 \text{ bits}$

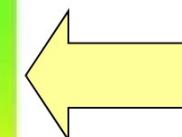
Color depth 24 bits
 = 16777216 colors



$(R = 0)$

$(G = 0)$

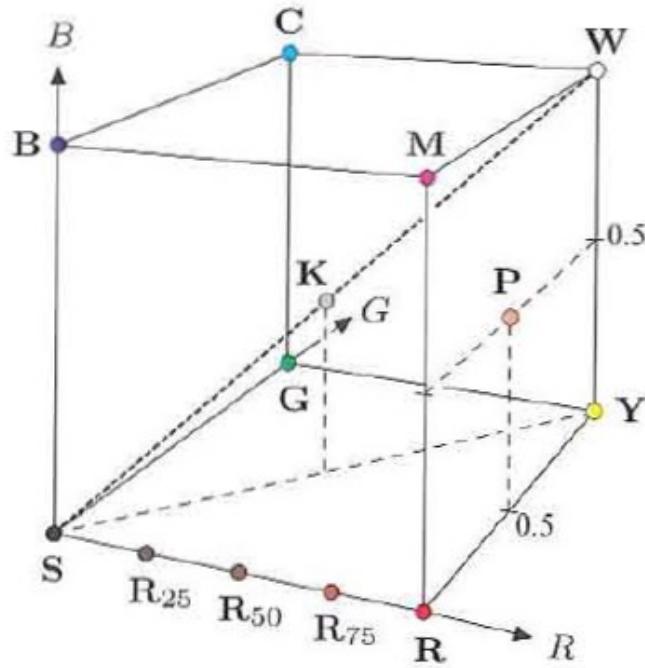
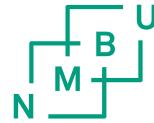
$(B = 0)$



Hidden faces
of the cube

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.)

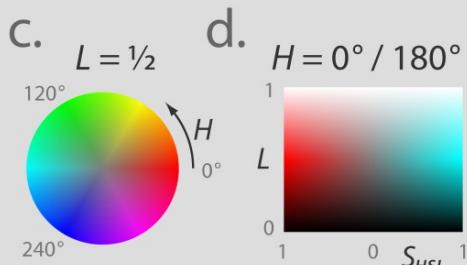
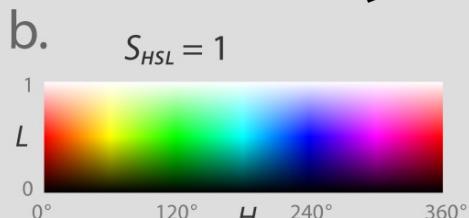
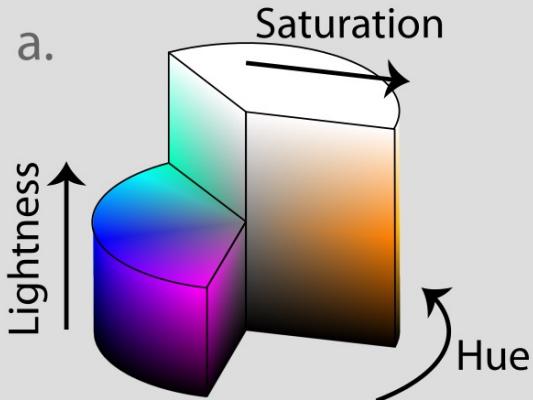
Colour space



| Point | Color | R | G | B |
|-----------------|----------|------|------|------|
| S | Black | 0.00 | 0.00 | 0.00 |
| R | Red | 1.00 | 0.00 | 0.00 |
| Y | Yellow | 1.00 | 1.00 | 0.00 |
| G | Green | 0.00 | 1.00 | 0.00 |
| C | Cyan | 0.00 | 1.00 | 1.00 |
| B | Blue | 0.00 | 0.00 | 1.00 |
| M | Magenta | 1.00 | 0.00 | 1.00 |
| W | White | 1.00 | 1.00 | 1.00 |
| K | 50% Gray | 0.50 | 0.50 | 0.50 |
| R ₇₅ | 75% Red | 0.75 | 0.00 | 0.00 |
| R ₅₀ | 50% Red | 0.50 | 0.00 | 0.00 |
| R ₂₅ | 25% Red | 0.25 | 0.00 | 0.00 |
| P | Pink | 1.00 | 0.50 | 0.50 |

Figure 8.1 Representation of the RGB color space as a three-dimensional unit cube. The primary colors red (R), green (G), and blue (B) form the coordinate system. The “pure” red color (R), green (G), blue (B), cyan (C), magenta (M), and yellow (Y) lie on the vertices of the color cube. All the shades of gray, of which K is an example, lie on the diagonal between black S and white W .

HSL



HSV

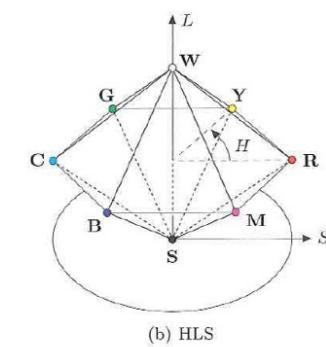
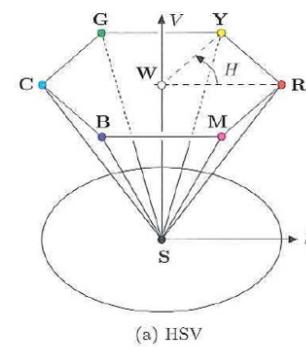
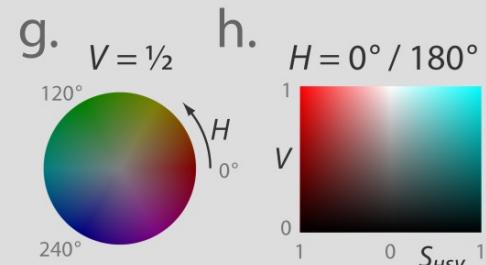
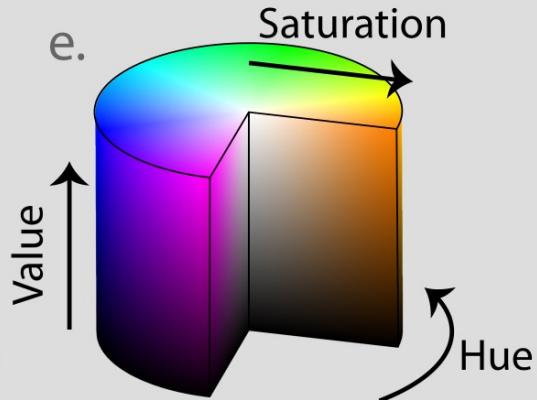
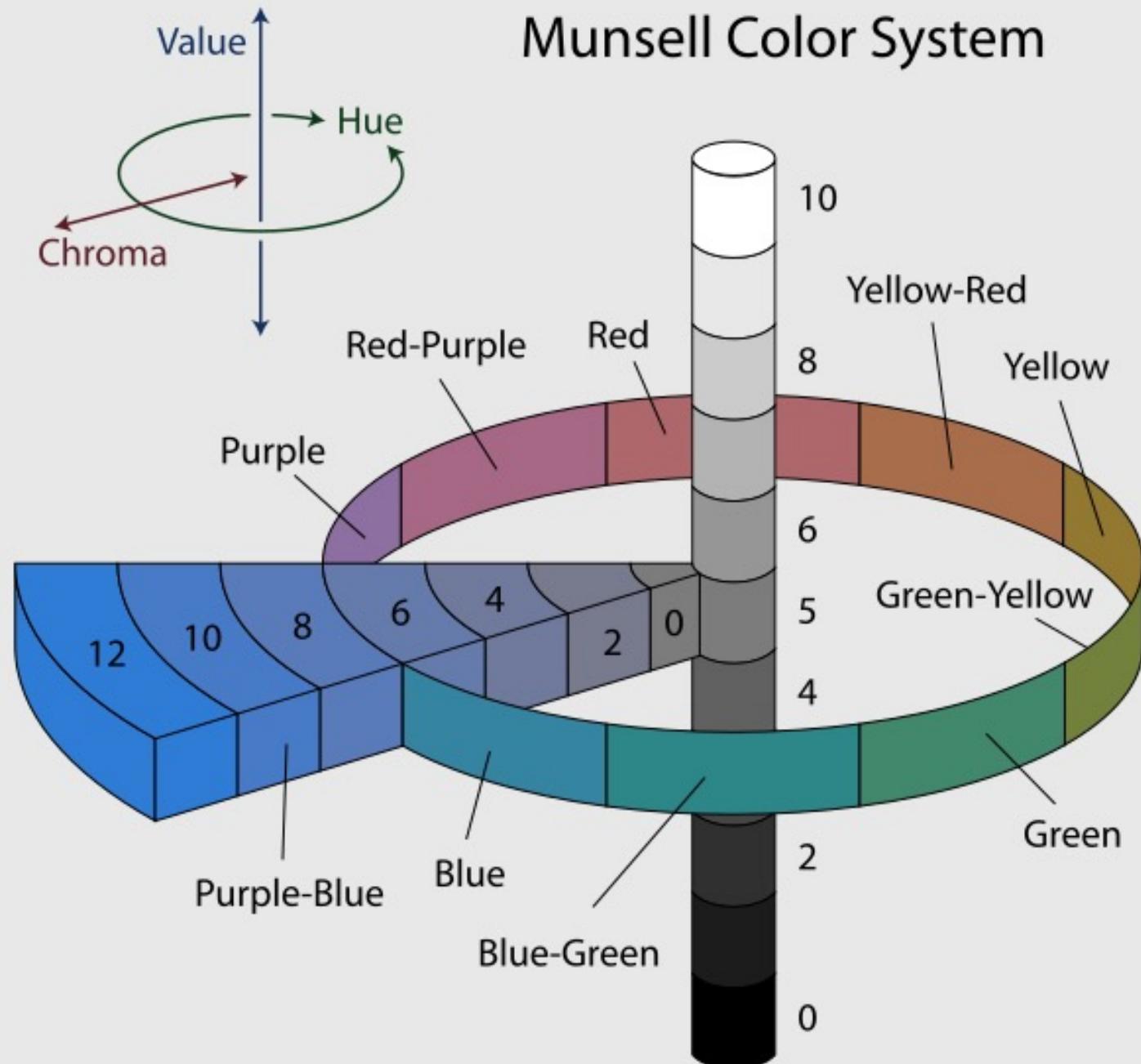


Figure 8.11 HSV and HLS color space are traditionally visualized as a single or double hexagonal pyramid. The brightness V (or L) is represented by the vertical dimension, the color saturation S by the radius from the pyramid's axis, and the hue h by the angle. In both cases, the primary colors red (R), green (G), and blue (B) and the mixed colors yellow (Y), cyan (C), and magenta (M) lie on a common plane with black (S) at the tip. The essential difference between the HSV and HLS color spaces is the location of the white point (W).

Munsell Color System

U
B
M



Distribution of colours in various spaces

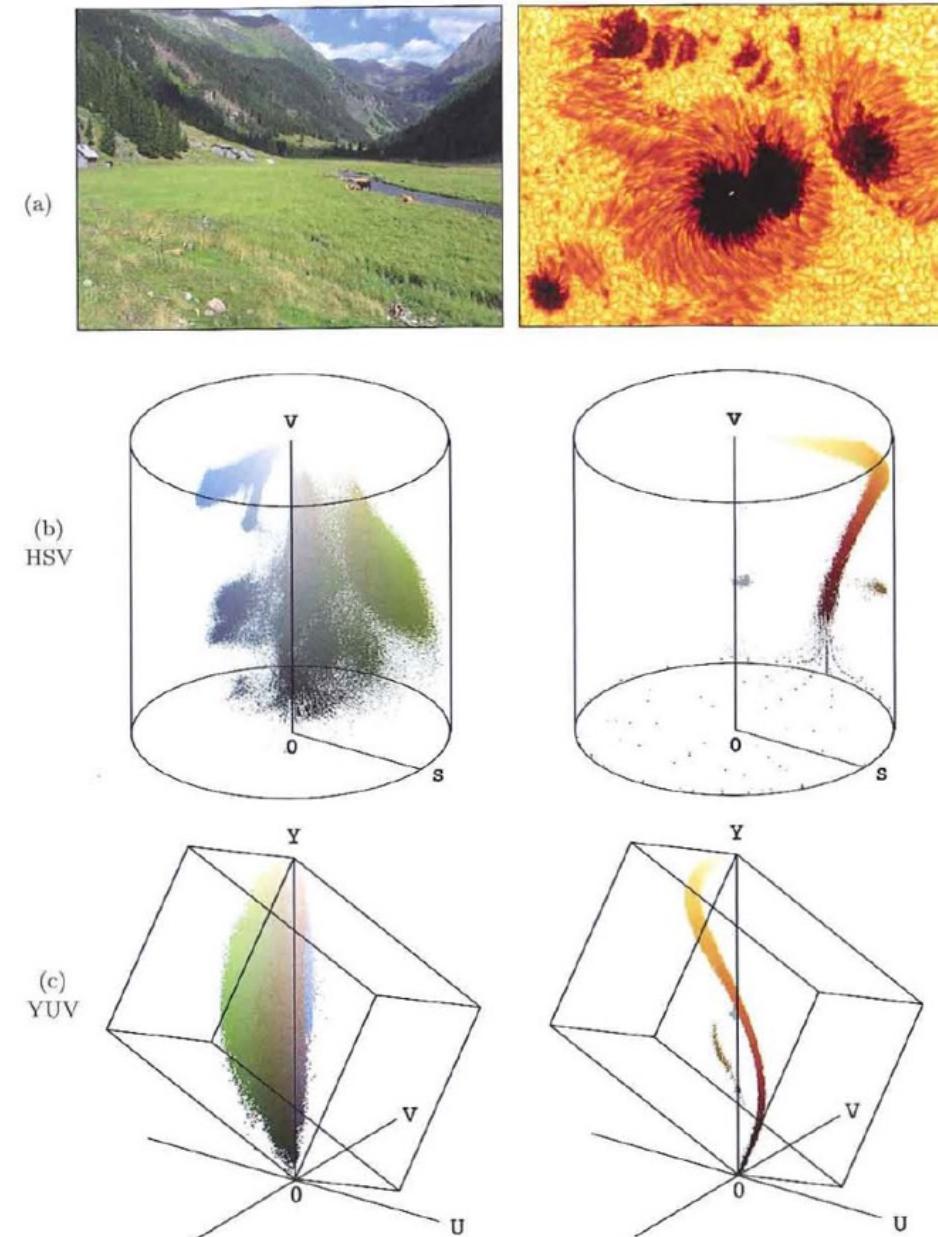
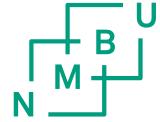
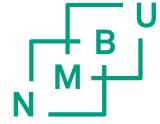


Figure 8.18 Examples of the color distribution of natural images in different color spaces. Original images (a); color distribution in HSV- (b), and YUV-space (c). See Fig. 8.9 for the corresponding distributions in RGB color space.



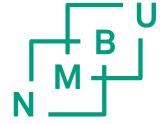
Color models

- RGB
 - The red, green, blue and cyan, magenta, yellow models are closely related, the primary colors of each form the secondary colors of the other. These are also the most representative models for additive and subtractive colors, respectively. RGB is also the basic color model for on-screen display.
- Munsell
 - The Munsell color system is one of the most influential systems developed for ordering colors that can be used for production. While its practical application is mostly outside of print production, it still forms the basis for most other work on color modeling.
- CIE
 - The CIE color models are highly influential systems for measuring color and distinguishing between colors. We will examine three CIE models: [CIEXYZ](#), [CIELUV](#), and [CIELAB](#). The last of these, CIELAB, is very important to color management
- NCS
 - Natural Color System. (Jotun)



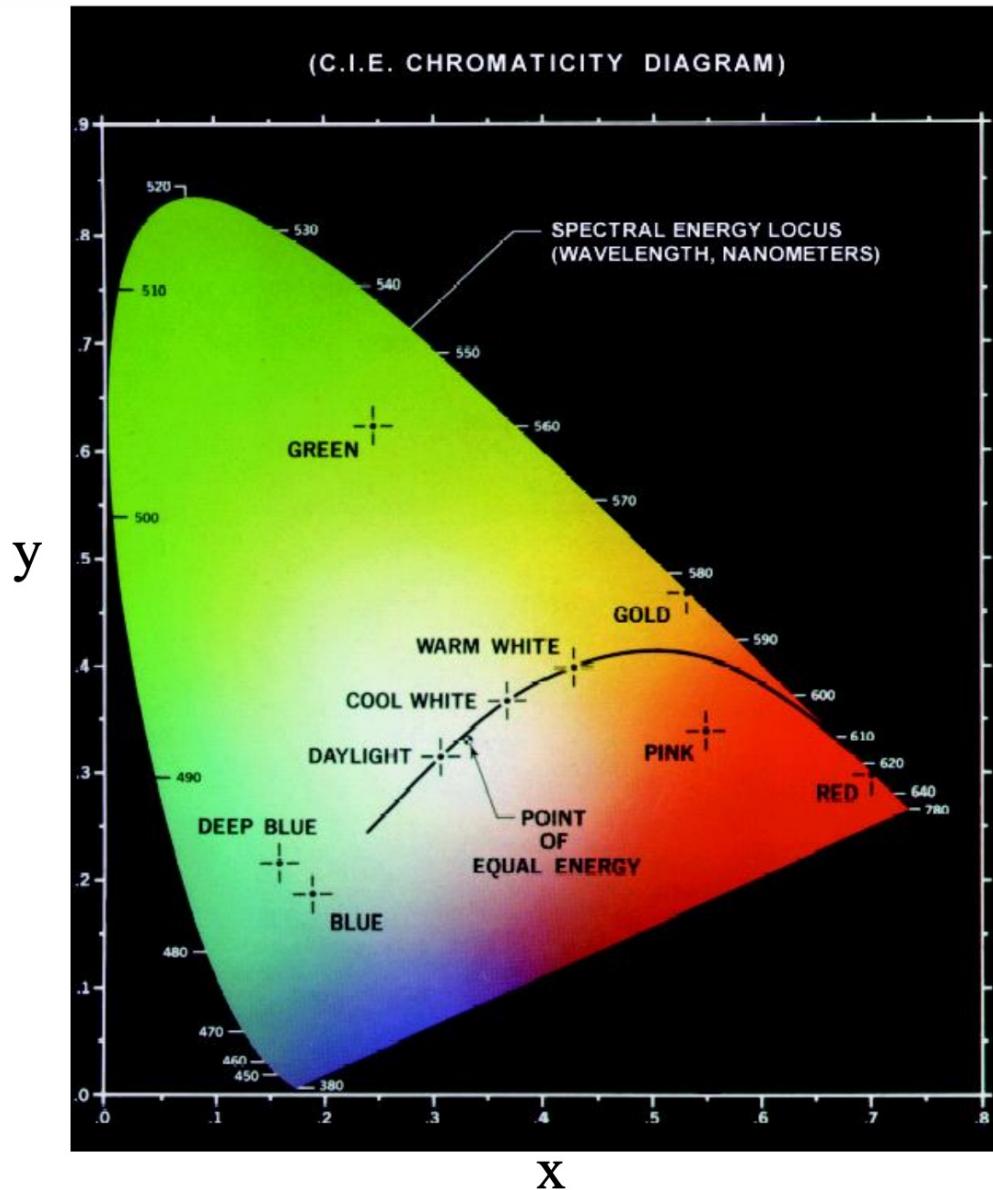
CIE

- CIE stands for Comission Internationale de l'Eclairage (International Commission on Illumination). The commission was founded in 1913 as an autonomous international board to provide a forum for the exchange of ideas and information and to set standards for all things related to lighting.
- As a part of this mission, CIE has a technical committee, Vison and Colour, that has been a leading force in colorimetry since it first met to set its standards in Cambridge, England, in 1931.



CIE

- Source A
 - A tungsten-filament lamp with a color temperature of 2854K
- Source B
 - A model of noon sunlight with a temperature of 4800
- Source C
 - A model of average daylight with a temperature of 6500K



Trichromatic coefficients:

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

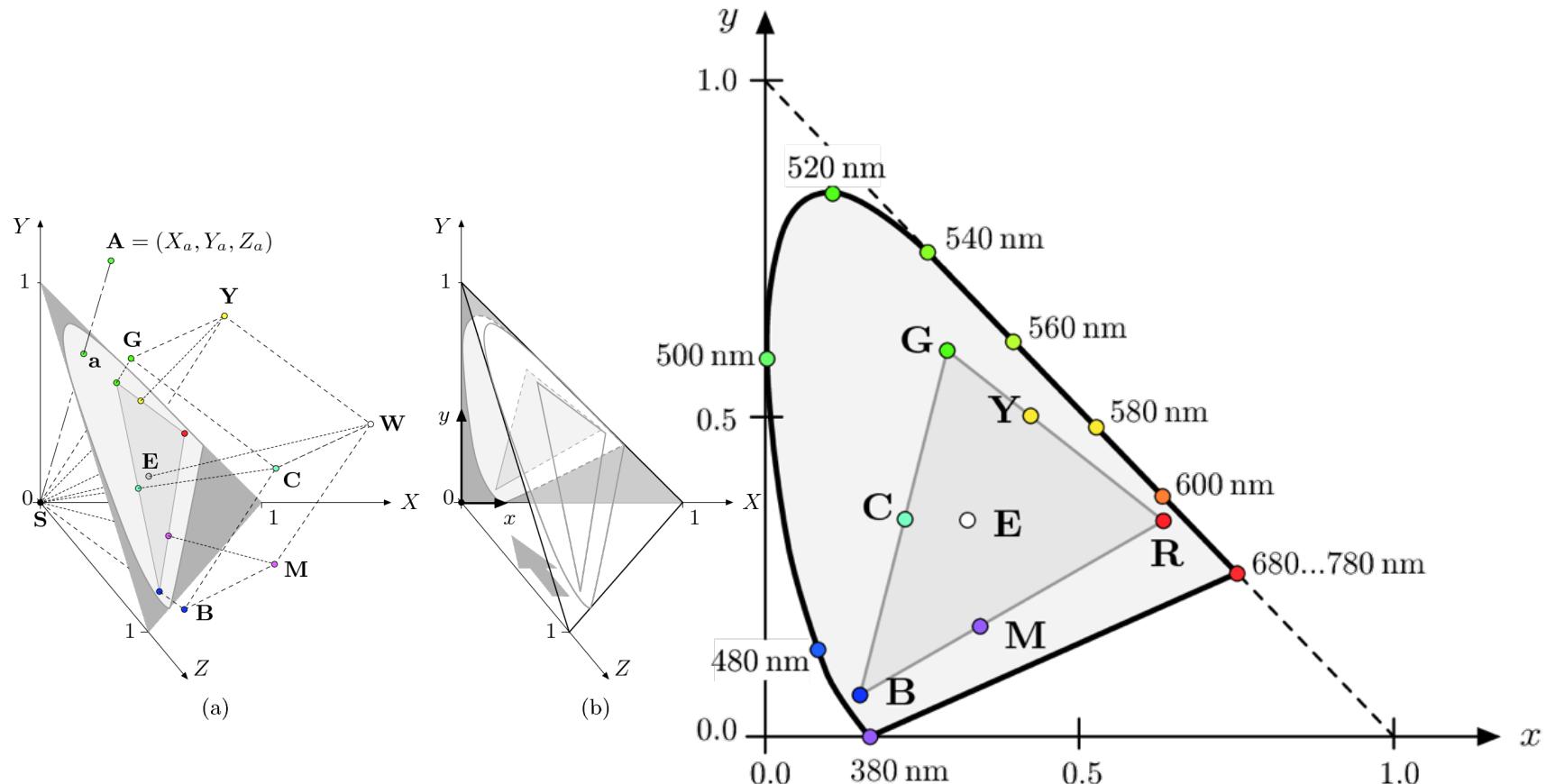
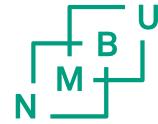
$$z = \frac{Z}{X + Y + Z}$$

$$x + y + z = 1$$

Points on the boundary are fully saturated colors

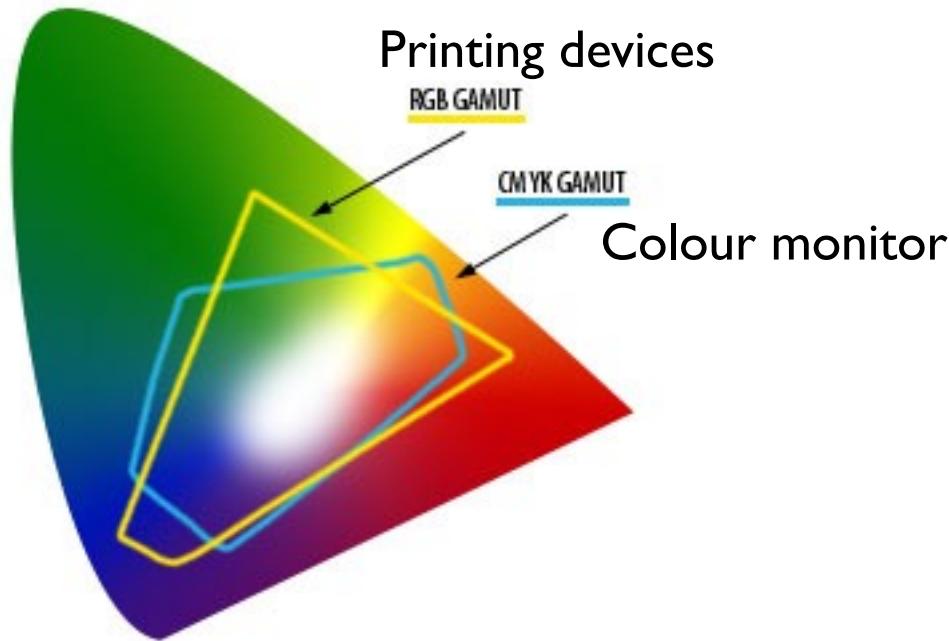
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.)

CIE XYZ colour room



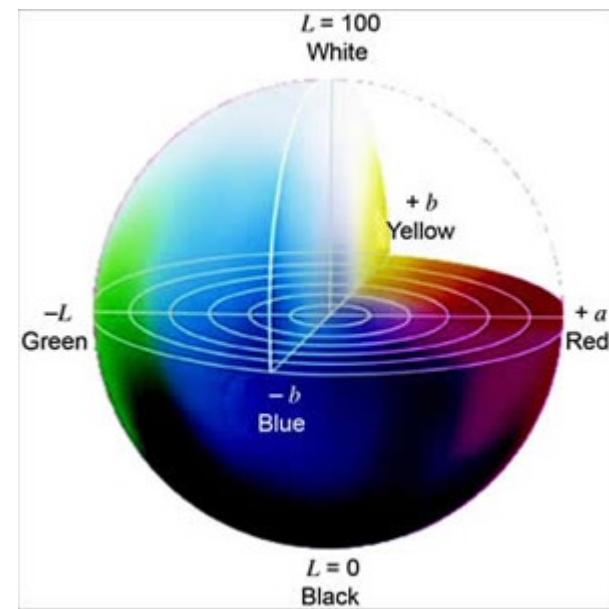
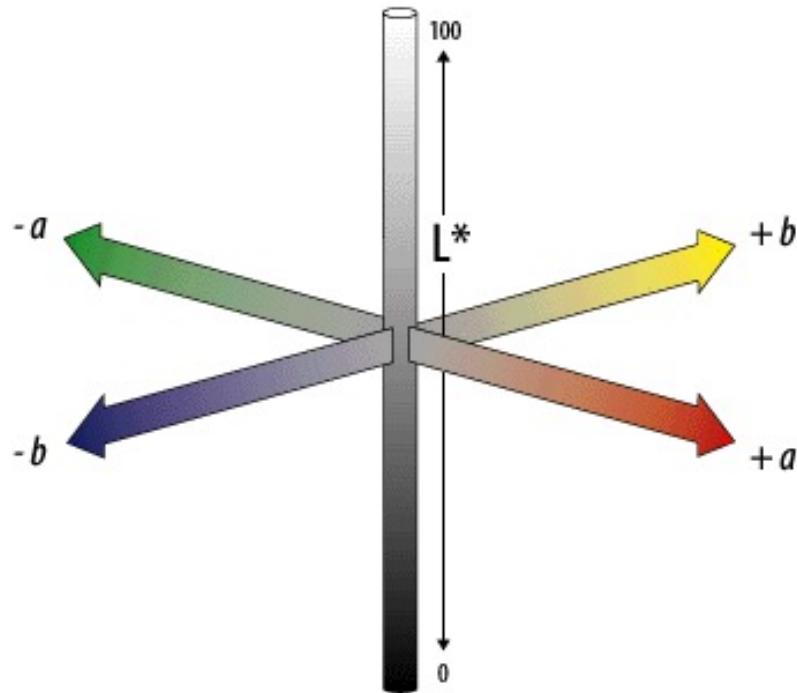
RGB color model

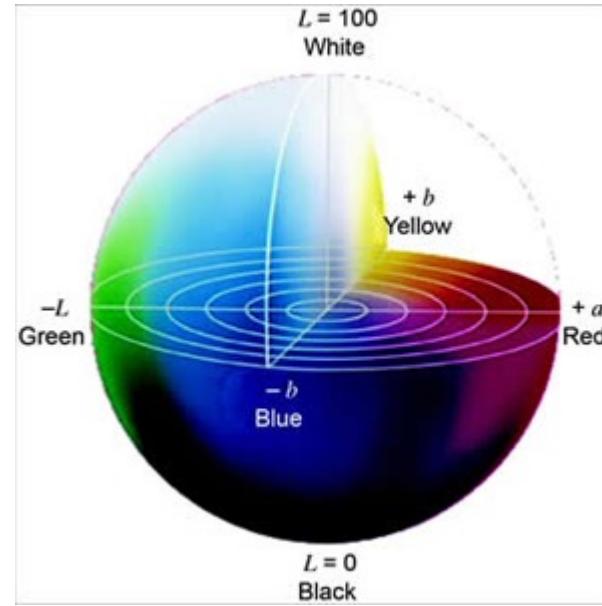
- Gamut constraint



<https://www.youtube.com/watch?v=O0nYJ0Mjx10>

CIE LAB color model



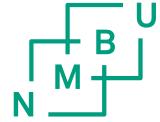


Delta E, ΔE:

Total colour difference ΔE^*_{ab} (Equation 1) from a reference colour (L^*_1, a^*_1, b^*_1) to a target colour (L^*_2, a^*_2, b^*_2) in the CIE Lab space is given by:

Equation 1: Colour-difference formula.

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2}$$



NCS (Natural Color System)

- Jotun paint
- Not direct transformations of RGB etc
- NCSCOLOUR.COM
 - <http://ncscolour.com/design/work-digitally-with-ncs/colouring-to-a-new-level/>
- EASYRGB.COM
 - <http://www.easyrgb.com/index.php?X=SEEK>

COLOUR LAB

- <https://www.ntnu.edu/colourlab#/view/about>



LAB used to describe colour of wooden facades

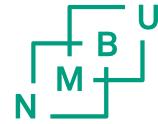
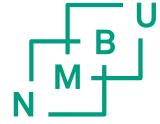


Figure 3: Visual appearance of façade in October 2013 (left) and October 2015 (right).

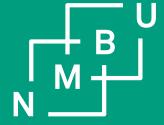
| Material Position | 0 | 1 | 4 | 6 | 9 | 12 | 17 | 23 | 29 | 36 |
|-------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| | Oct 2013 | Dec 2013 | Mar 2014 | May 2014 | Aug 2014 | Oct 2014 | Apr 2015 | Oct 2015 | Apr 2016 | Oct 2016 |
| north | | | | | | | | | | |
| north m | | | | | | | | | | |
| south | | | | | | | | | | |
| south m | | | | | | | | | | |
| east | | | | | | | | | | |
| east m | | | | | | | | | | |
| west | | | | | | | | | | |
| west m | | | | | | | | | | |
| deck | | | | | | | | | | |
| deck m | | | | | | | | | | |

PA

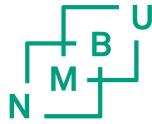
You need to know these terms



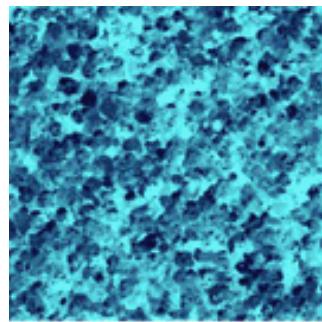
- RGB colour space
- CIE
- CIE Lab colour space
- Hue
- Chroma
- Brightness
- Lightness
- Gamut



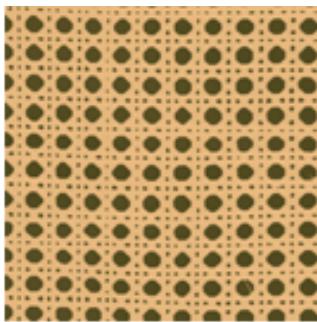
Texture analysis.



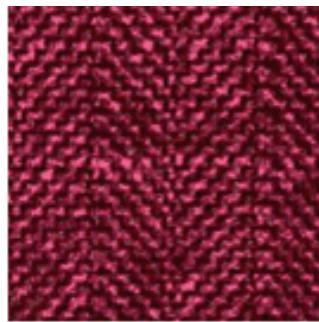
Textures



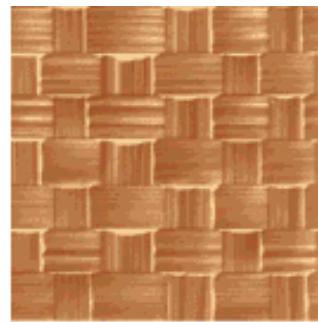
D28



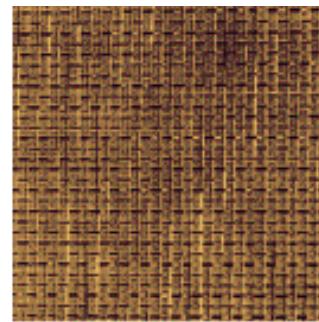
D101



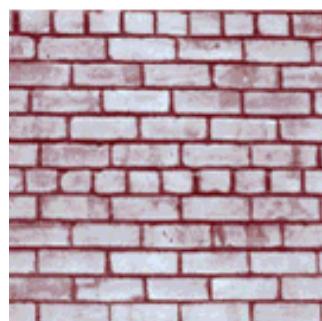
D17



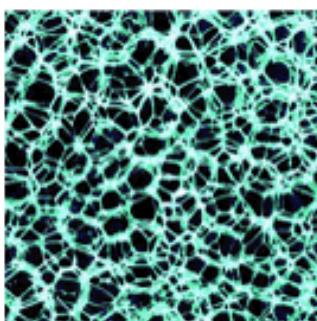
D64



D14



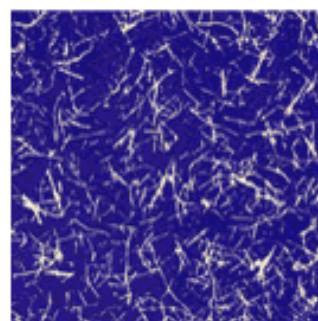
D95



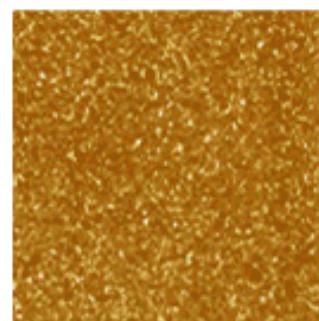
D111



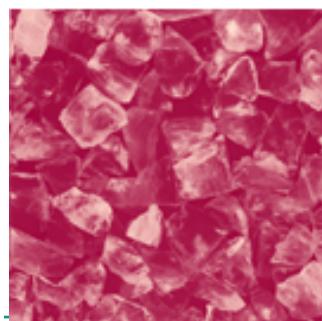
D44



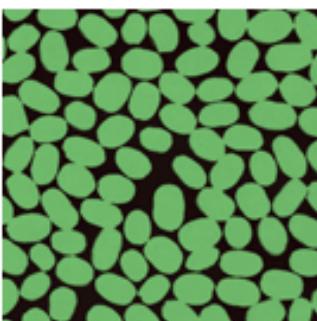
D109



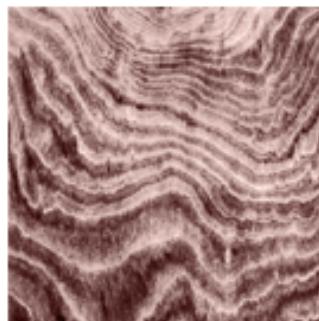
D32



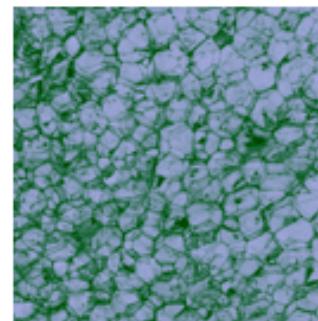
D99



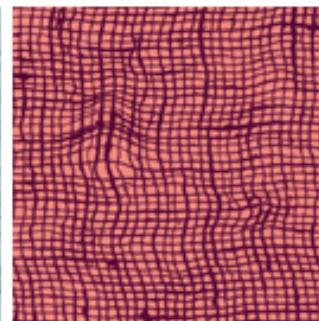
D75



D71



D112



D104

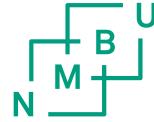


Image Texture

- Texture analysis refers to the characterization of regions in an image by their texture content
- Texture analysis attempts to quantify intuitive qualities described by terms such as rough, smooth, silky, or bumpy as a function of the spatial variation in pixel intensities
- In this sense, the roughness or bumpiness refers to variations in the intensity values, or gray levels

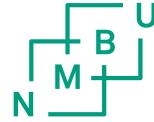


Image Texture

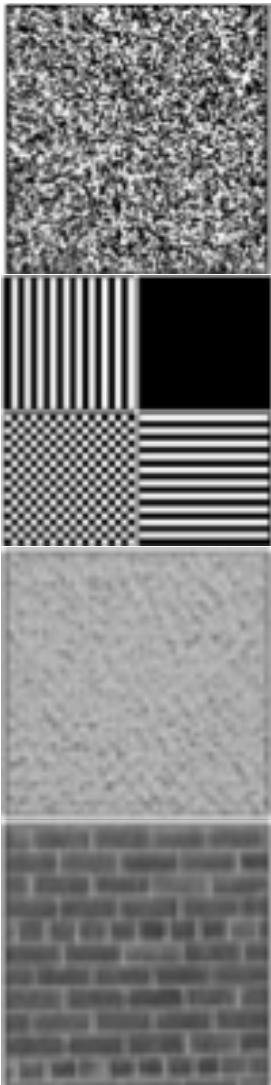
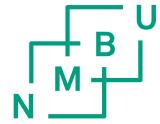
- Texture analysis is used in a variety of applications, including remote sensing, automated inspection, and medical image processing
- Texture analysis can be used to find the texture boundaries, called texture segmentation
- Texture analysis can be helpful when objects in an image are more characterized by their texture than by intensity, and traditional thresholding techniques cannot be used effectively



Features

- Image features can be found from:
 - Edges
 - Gives the borders between image regions (sometimes incomplete)
 - Homogenous regions
 - Mean and variance are useful for describing them
 - Texture of a local sliding window

Visual Texture

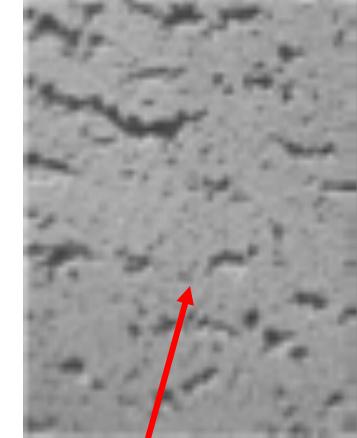
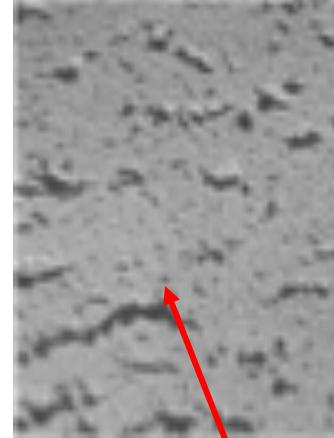


Stochastic
(isotropic)

Deterministic
(anisotropic)

Semi Stochastic

Semi Deterministic



Visual Texture
Perception

Brodatz Textures

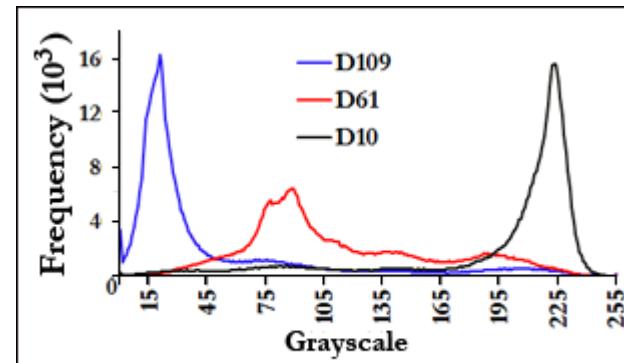
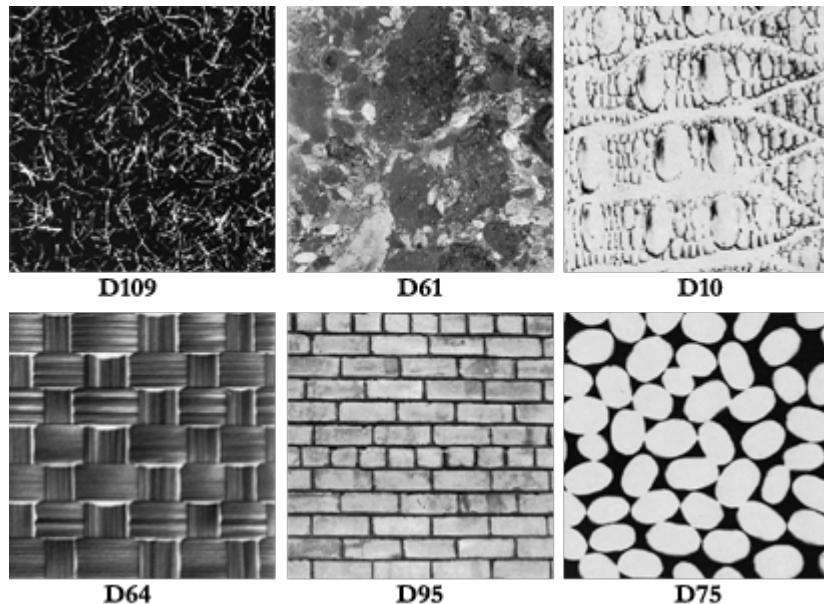
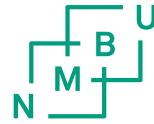


Figure 1

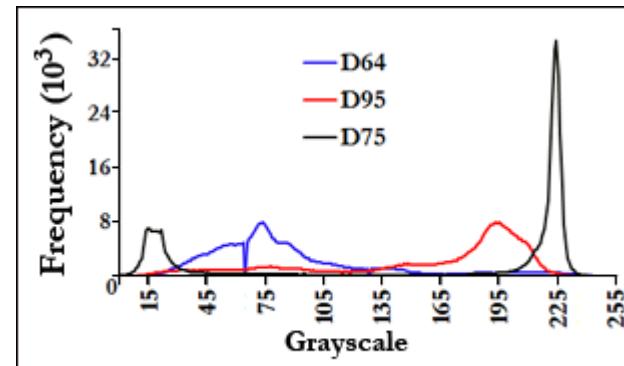
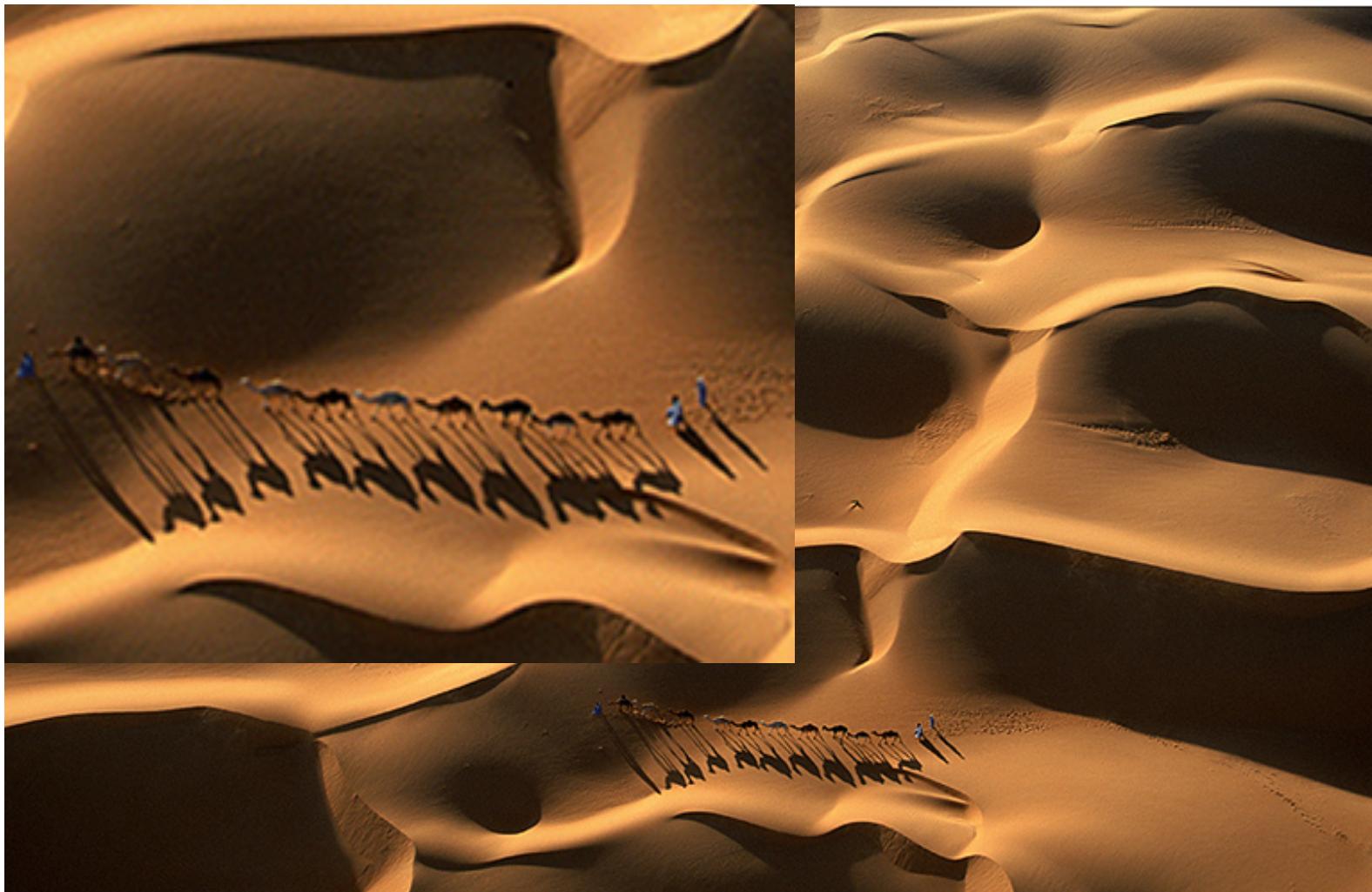
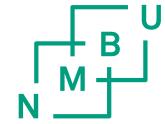


Figure 2

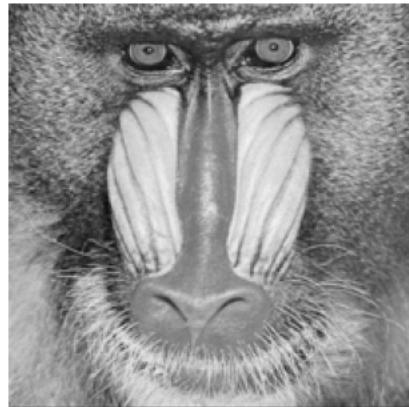
Brodatz album contains 112 texture images

http://multibandtexture.recherche.usherbrooke.ca/original_brodatz.html

Texture analysis: A matter of scale



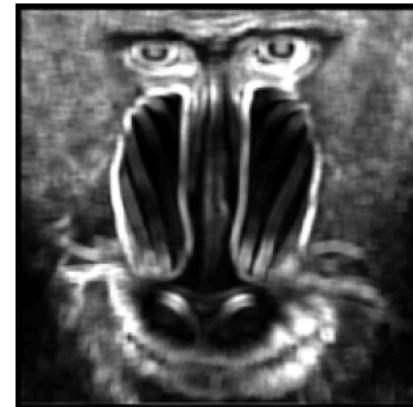
Example of scale dependence



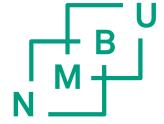
Original image



Variance feature
computed in window
of size 3x3



Variance feature
computed in window
of size 15x15



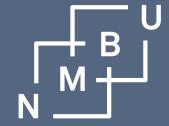
Texture feature extraction

Histograms

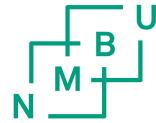
GLCM – Grey Level Co-occurrence matrix



Histograms



First order statistics from histograms



- Mean (hardly a useful feature)

$$\mu = \frac{\sum_{i=0}^{G-1} ip(i)}{\sum_{i=0}^{G-1} p(i)} = \frac{\sum_{i=0}^{G-1} ip(i)}{n} = \sum_{i=0}^{G-1} iP(i)$$

- Variance (a more credible feature, measures region "roughness")

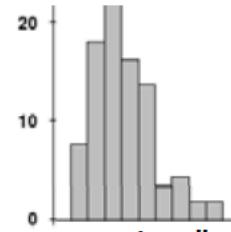
$$\sigma^2 = \sum_{i=0}^{G-1} (i - \mu)^2 P(i)$$

- Skewness (are the texel intensities usually darker/lighter than average?)

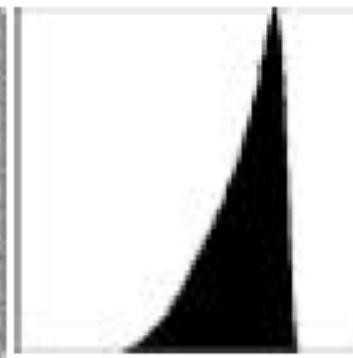
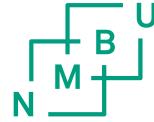
$$\gamma_3 = \frac{1}{\sigma^3} \sum_{i=0}^{G-1} (i - \mu)^3 P(i) = \frac{m_3}{\sigma^3}$$

- Kurtosis (how "peaked" is the graylevel distribution?)

$$\gamma_4 = \frac{1}{\sigma^4} \sum_{i=0}^{G-1} (i - \mu)^4 P(i) - 3 = \frac{m_4}{\sigma^4} - 3$$



Example of first order statistics



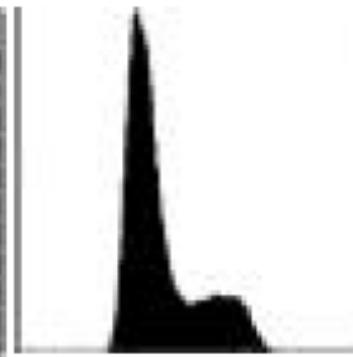
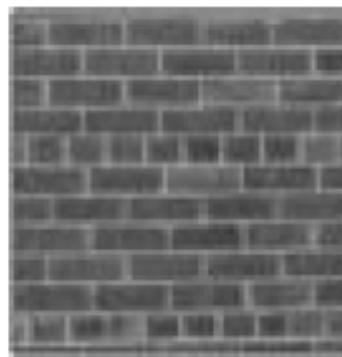
Mean 167.9

Variance 669.0

CV 0.15

Skewness -0.82

Curtosis 0.01



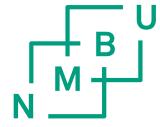
Mean 105.1

Variance 720

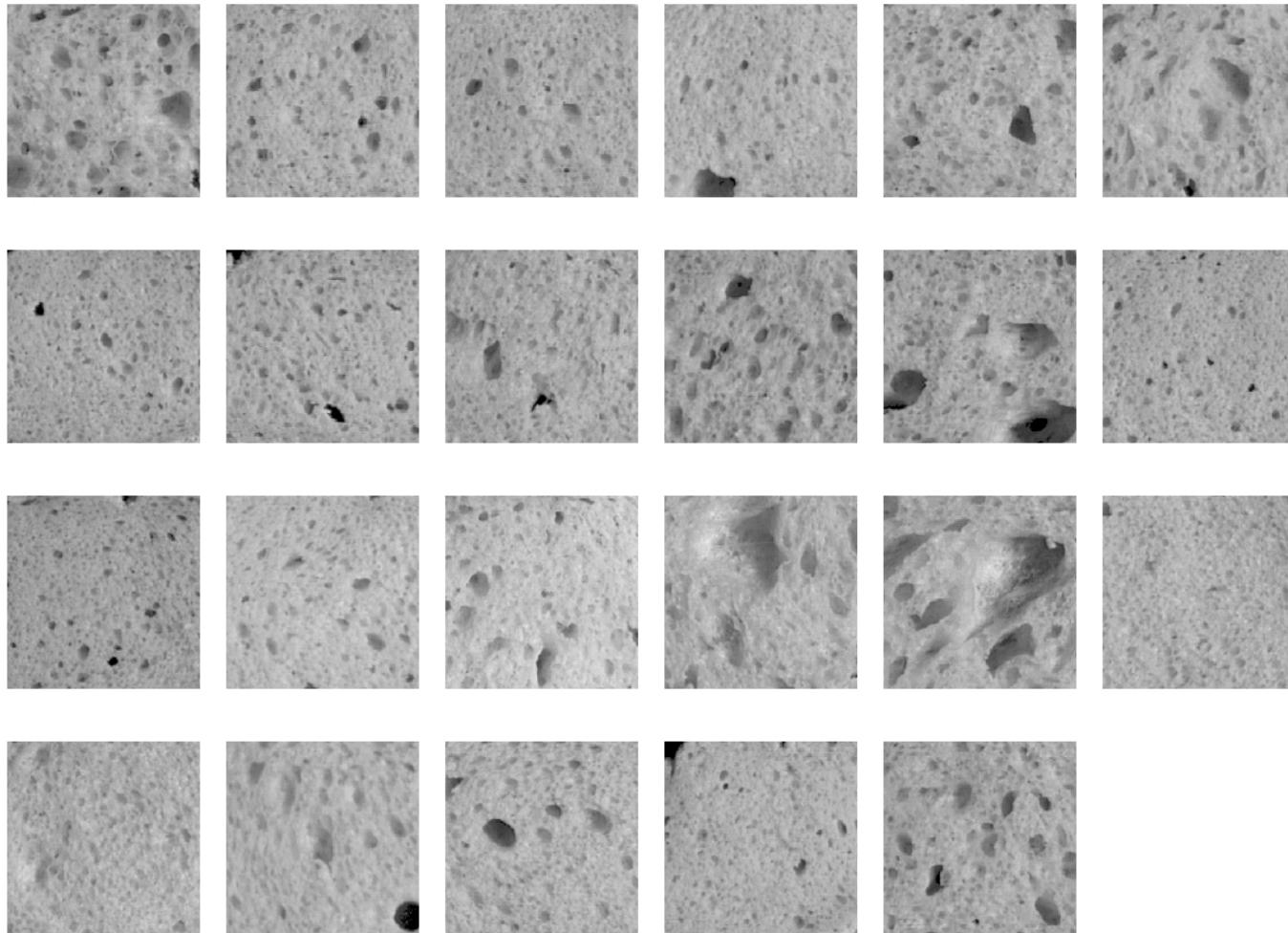
CV 0.26

Skewness 1.15

Curtosis 0.30

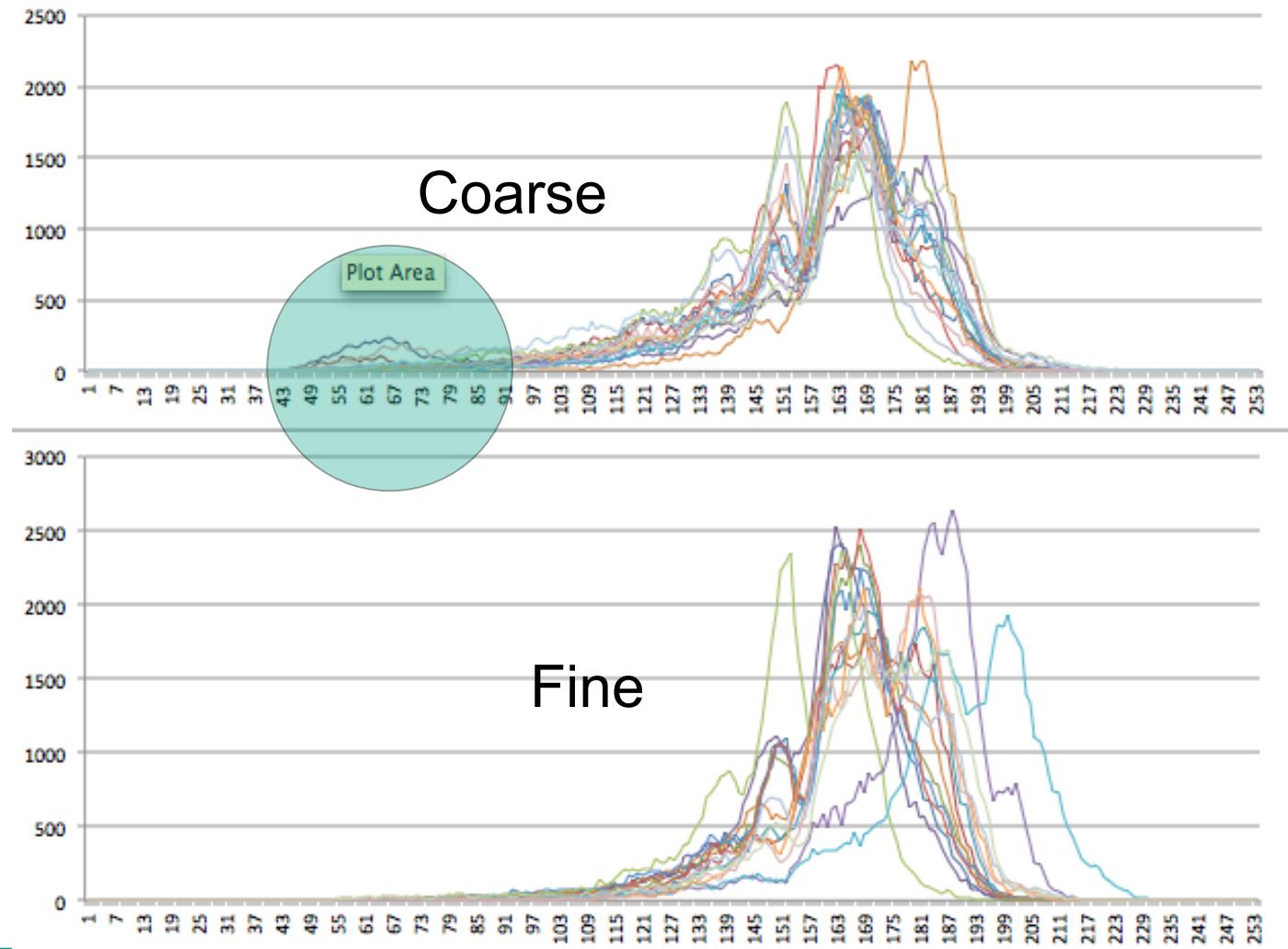


Baguette textures

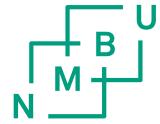


Baguette textures and histograms

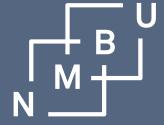
N M B U



Limitation with first order statistics

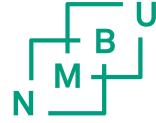


- Edges around objects are exaggerated
- 1. order statistics can not describe geometry or context
 - Can not discriminate between The image shows two groups of vertical black bars. The left group has five bars of equal height. The right group has three bars, with the middle one being significantly taller than the others.
 - Solution:
 - Calculate 1. order statistics with different resolutions, obtain information about higher order statistics
 - Use 2. or higher order statistics

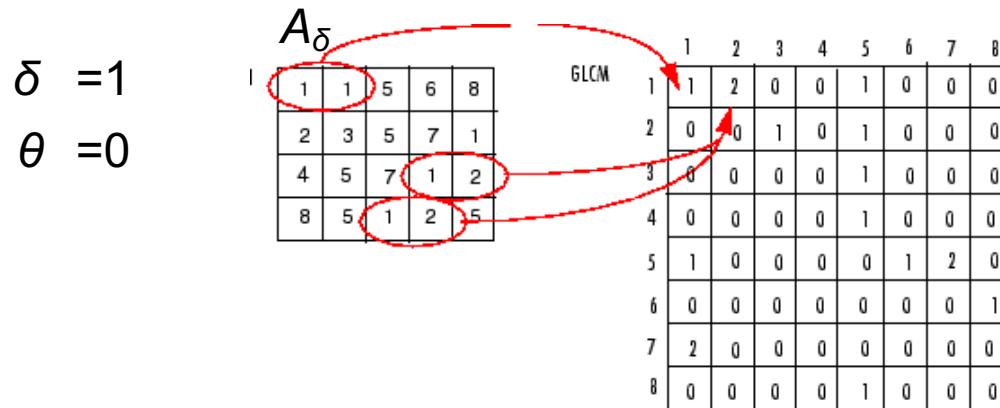


Grey Level Co-occurrence Matrix (GLCM)

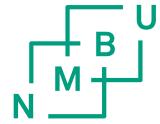
The gray level co-occurrence matrix (GLCM)



- Statistical calculations on the second-order histograms of the gray scale images.
- Calculate how often two pixels, in the matrix element $A_{\delta}(i, j)$, with intensity values i and j at a particular displacement distance δ from along a given direction θ (horizontally, vertically, or diagonally) occurs in the image



The GLCM matrix P



| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 2 | 4 |
| 5 | 3 | 2 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 1 | 1 | 5 | 2 | 1 |
| 2 | 5 | 4 | 1 | 3 |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 2 | 2 | 1 | 1 |
| 2 | 4 | 0 | 1 | 1 |
| 3 | 0 | 3 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 |

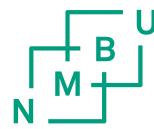
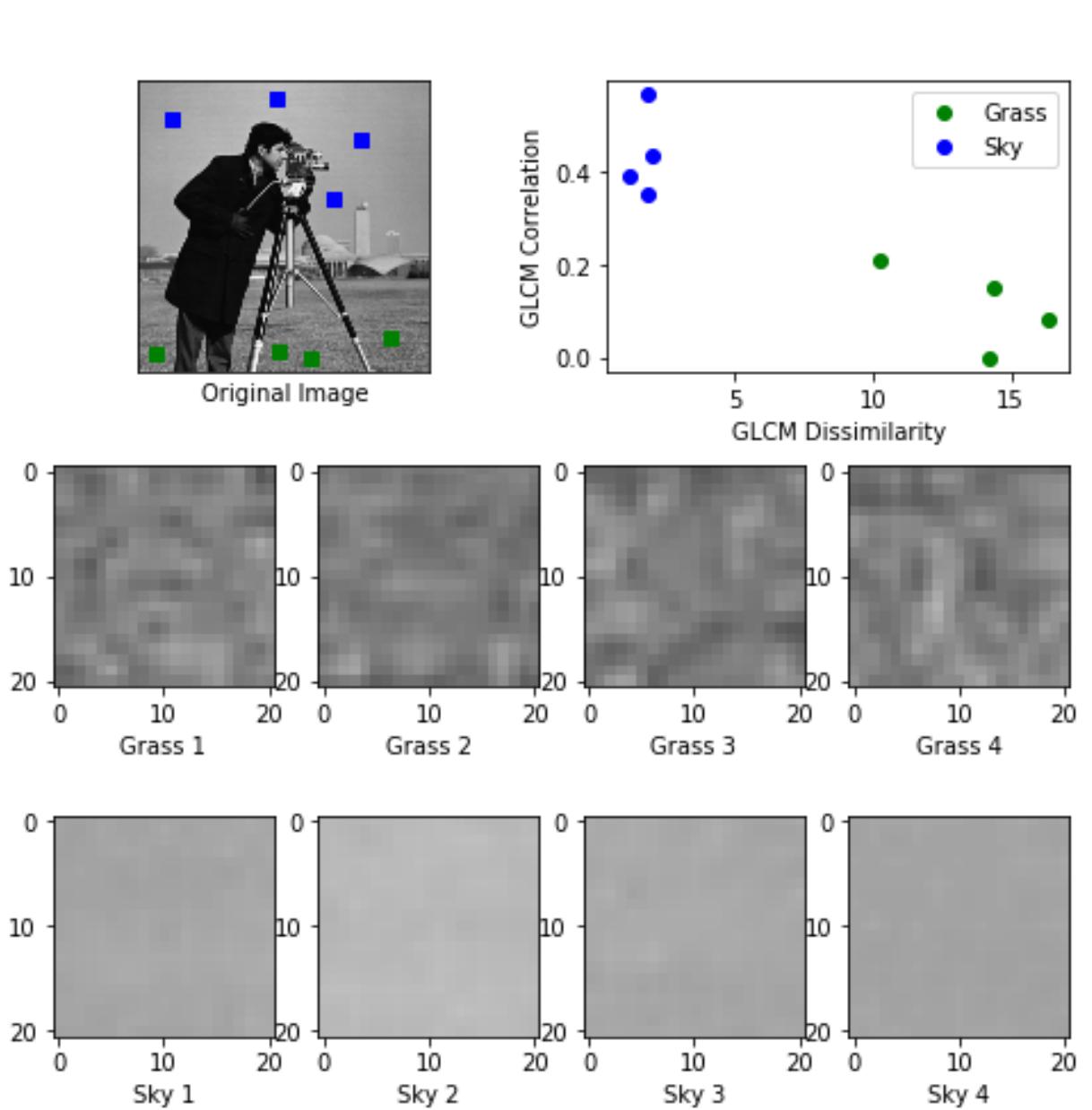
GLCM (0°)

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 2 | 4 |
| 5 | 3 | 2 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 1 | 1 | 5 | 2 | 1 |
| 2 | 5 | 4 | 1 | 3 |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 3 | 1 | 0 | 1 |
| 2 | 3 | 1 | 0 | 0 |
| 3 | 0 | 0 | 2 | 0 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 2 | 0 | 0 |

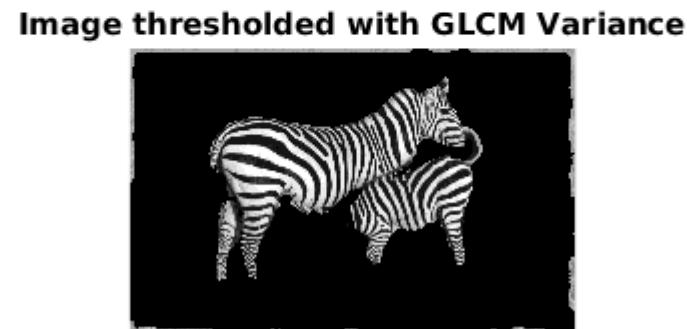
GLCM (45°)

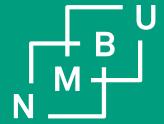
Grey level co-occurrence matrix features



Computing texture images

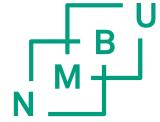
- Select a window size and select feature
- For each pixel compute the texture feature similar to filtering
- For each pixel, compute a local homogeneity measure in a local window





INF250

Multivariate analysis



Exam

Histograms:

- definitions, histogram binning, cumulative histogram, purpose of histogram equalization
- first order statistics, how is it calculated

Point operations:

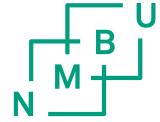
- contrast, brightness, inversion, thresholding – Otsu method

Filter:

- linear non-linear filters, purpose and building
- edge detection filters, canny, sobel, prewitt

Morphology:

- dilation, erosion etc, definitions + how to generate it
-



Exam

Objects:

- Run Length Encoding
- Geometric features from image
- Watershed segmentation

Colour images:

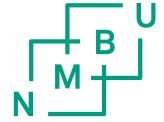
- colour spaces and colour models
- additive and subtractive colours

Texture:

- GLCM!!!

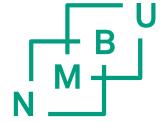
PCA:

- loading, score matrix
-



Topics for INF250

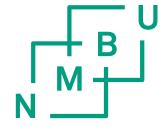
- Multivariate analysis
- PCA
- Clustering



The world is multivariate

Multivariate analysis refers to any statistical technique used to analyze data that arises from more than one variable

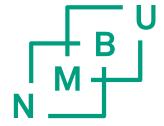


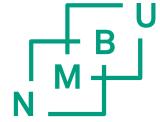


Who is this ?

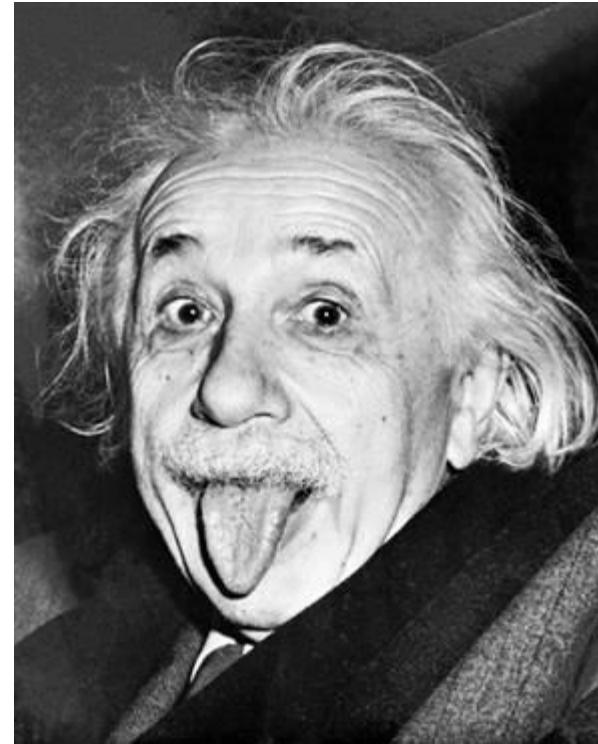


WHO IS THIS?

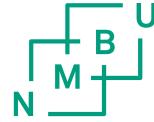




WHO IS THIS



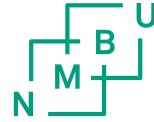
- The brain do not collect information from single parts, but for elements in a pattern
 - We need multivariate techniques!
-



Collecting data

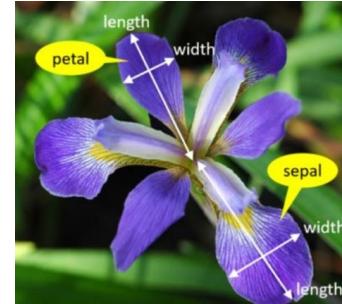
- Data are often collected as sensor input (X) and actual chemical/physical/sensory measurements as (Y)

| X-variables | | | | | | p | Y-variables | | | | | | q |
|-------------|-----------------|-----------------|-----|-----|-----------------|-----------------|-----------------|-----------------|-----------------|--|--|--|-----------------|
| Objects | X ₁₁ | X ₁₂ | ... | ... | X _{1p} | | Y ₁₁ | ... | Y _{1q} | | | | |
| | X ₂₁ | X ₂₂ | ... | ... | | Y ₂₁ | ... | Y _{2q} | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| | • | • | | | • | • | | | | | | | |
| n | X _{n1} | ... | | | X _{np} | | n | Y _{n1} | | | | | Y _{nq} |



Data presentation and modelling

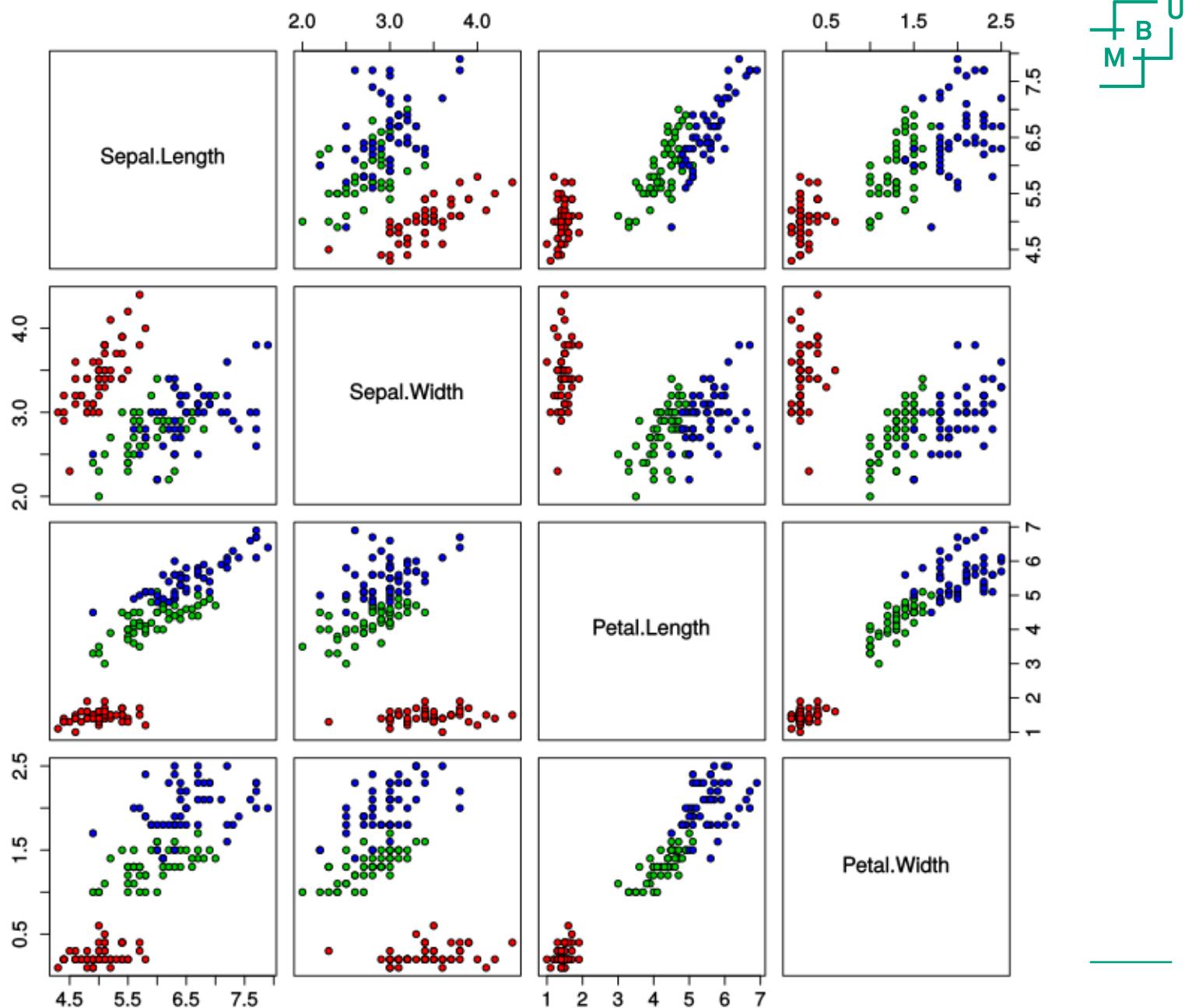
- One, Two, Three dimensional plots?
- Do we need a N dimension space to view data?
- How to find the optimal low dimension space that reveals maximum useful information?
- How to find the «not so useful information»
 - «NOISE»



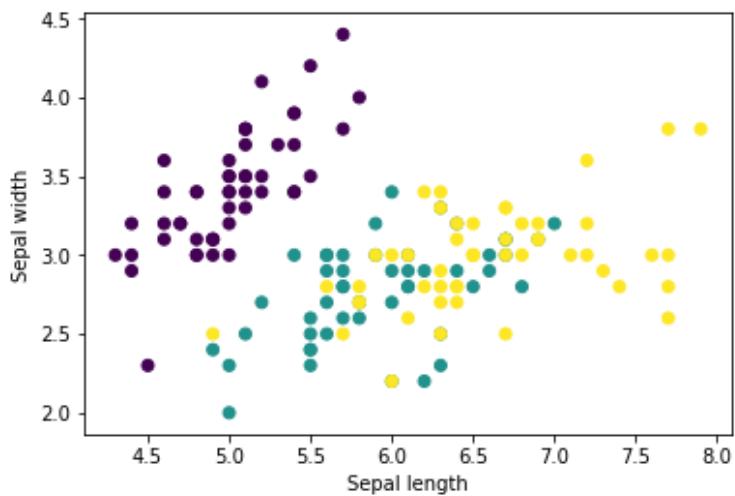
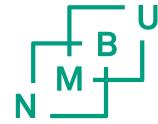
The Iris data set

- https://en.wikipedia.org/wiki/Iris_flower_data_set
- Often used as a demo dataset for multivariate analysis
- 3 species of Iris determined from 4 features: length and width of sepal and petals
- http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

Iris Data (red=setosa,green=versicolor,blue=virginica)



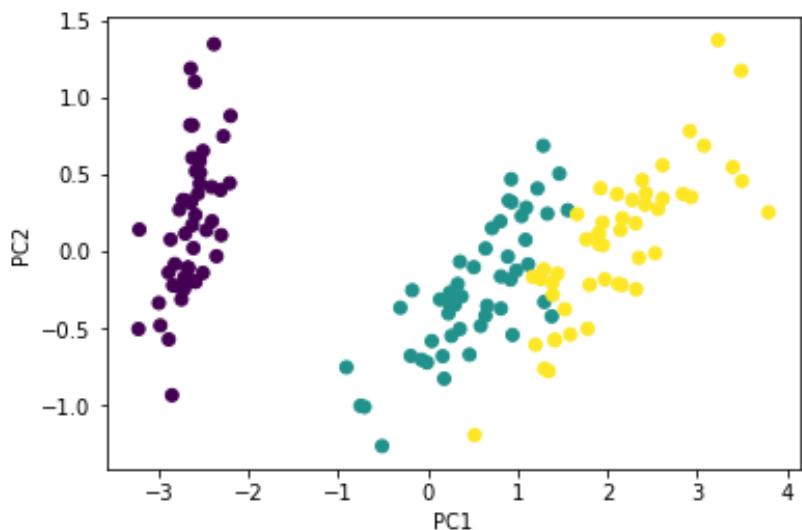
Principal Component Analysis



```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA

# import the Iris data
iris = datasets.load_iris()
X = iris.data
y = iris.target

plt.figure()
plt.scatter(X[:,0],X[:,1],c=y)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

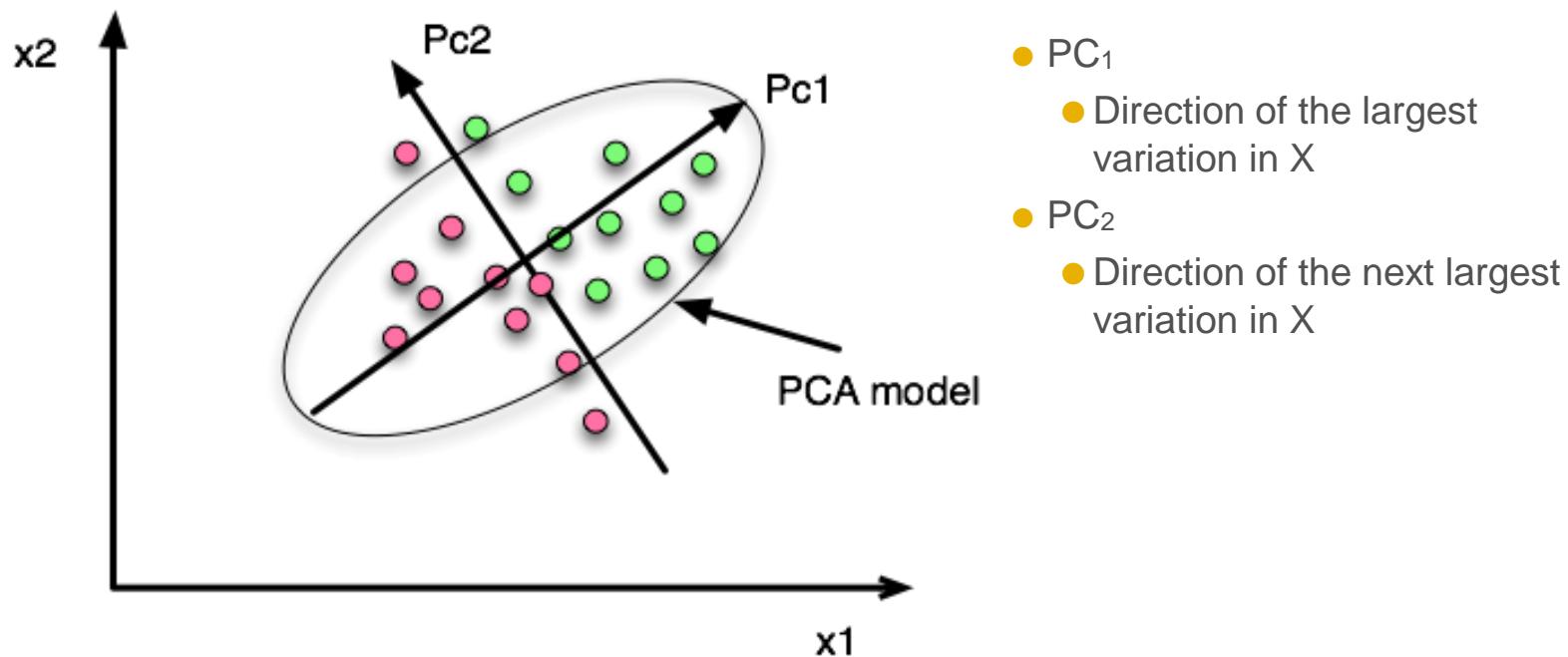


```
#simple PCA

X_reduced = PCA(n_components=3).fit_transform(iris.data)
plt.figure()
plt.scatter(X_reduced[:,1],X_reduced[:,2],c=y)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

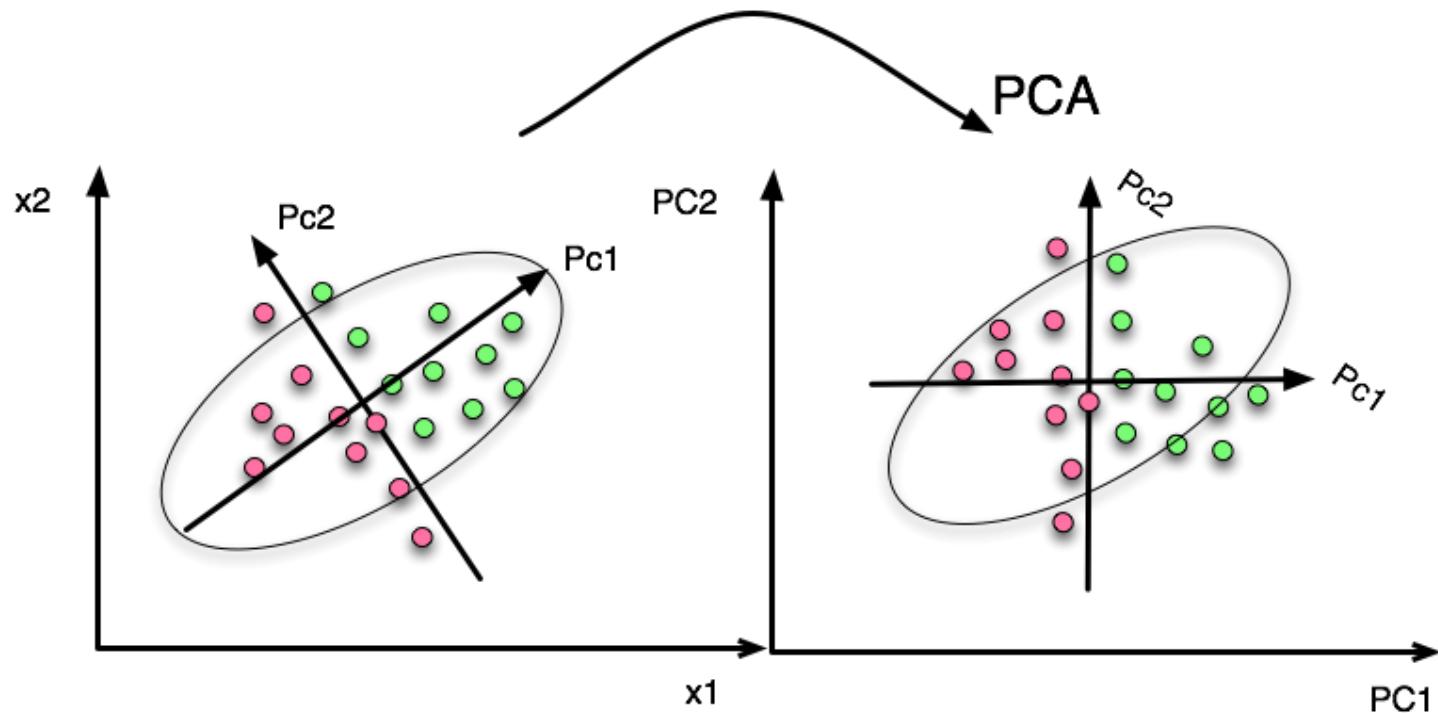
GRAPHICAL INTERPRETATION OF PCA

- The main idea is to form a minimum number of new variables that describes the variation of the data by a linear combination of the original values

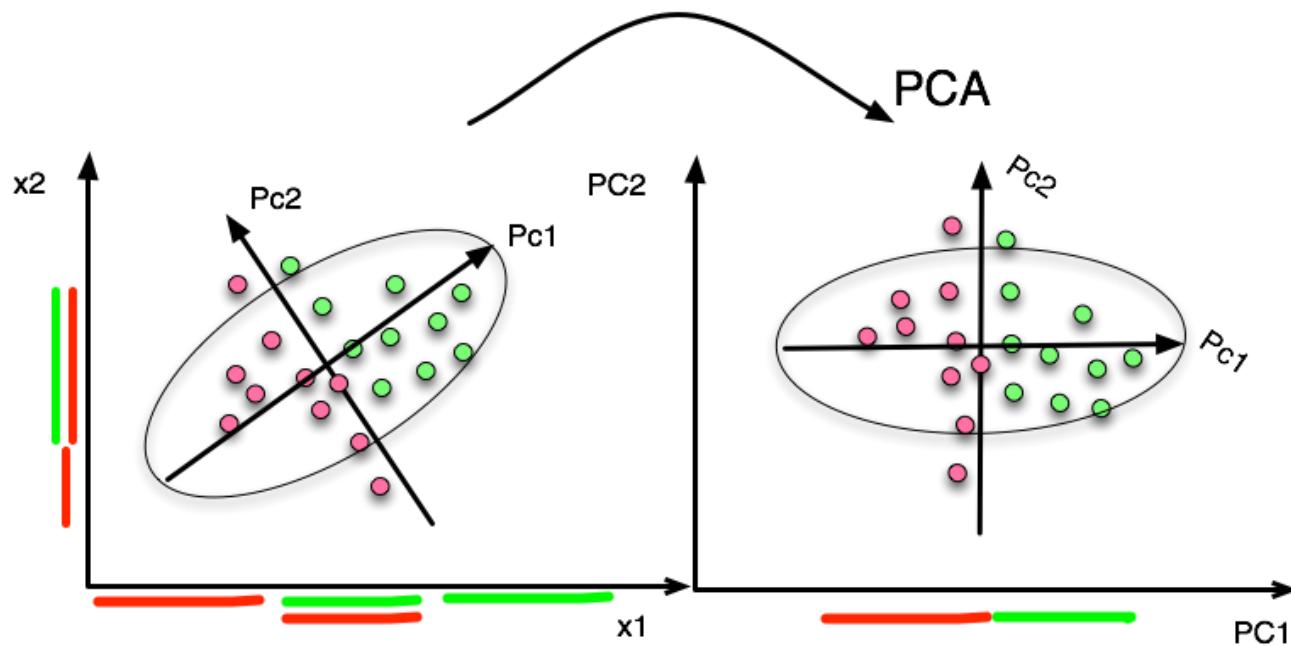


The Principal Components

- The new coordinate system represents a stepwise rotation along the principal orthogonal axes

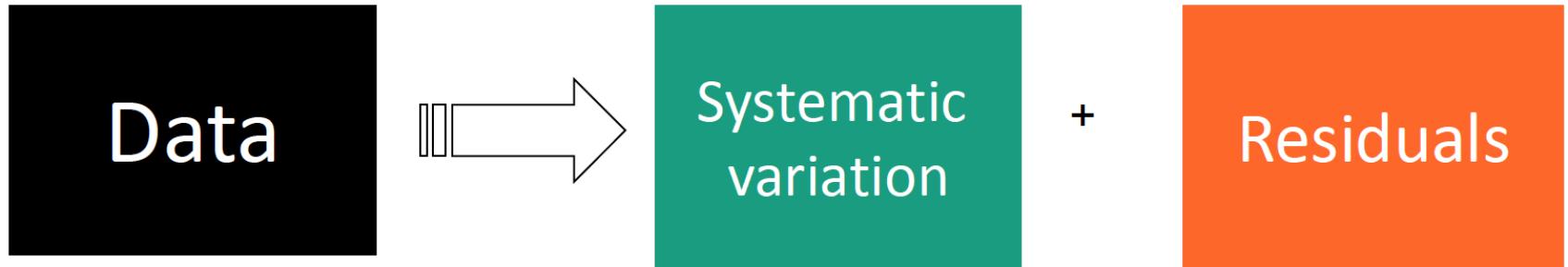
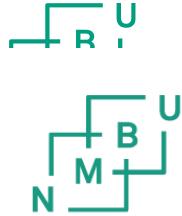


USE PCA TO SEPARATE CLASSES

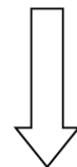


Separation of classes with a rotation

PCA basics



$$X = TP' + E$$



X : data matrix

T : PCA scores matrix

P : PCA loadings matrix

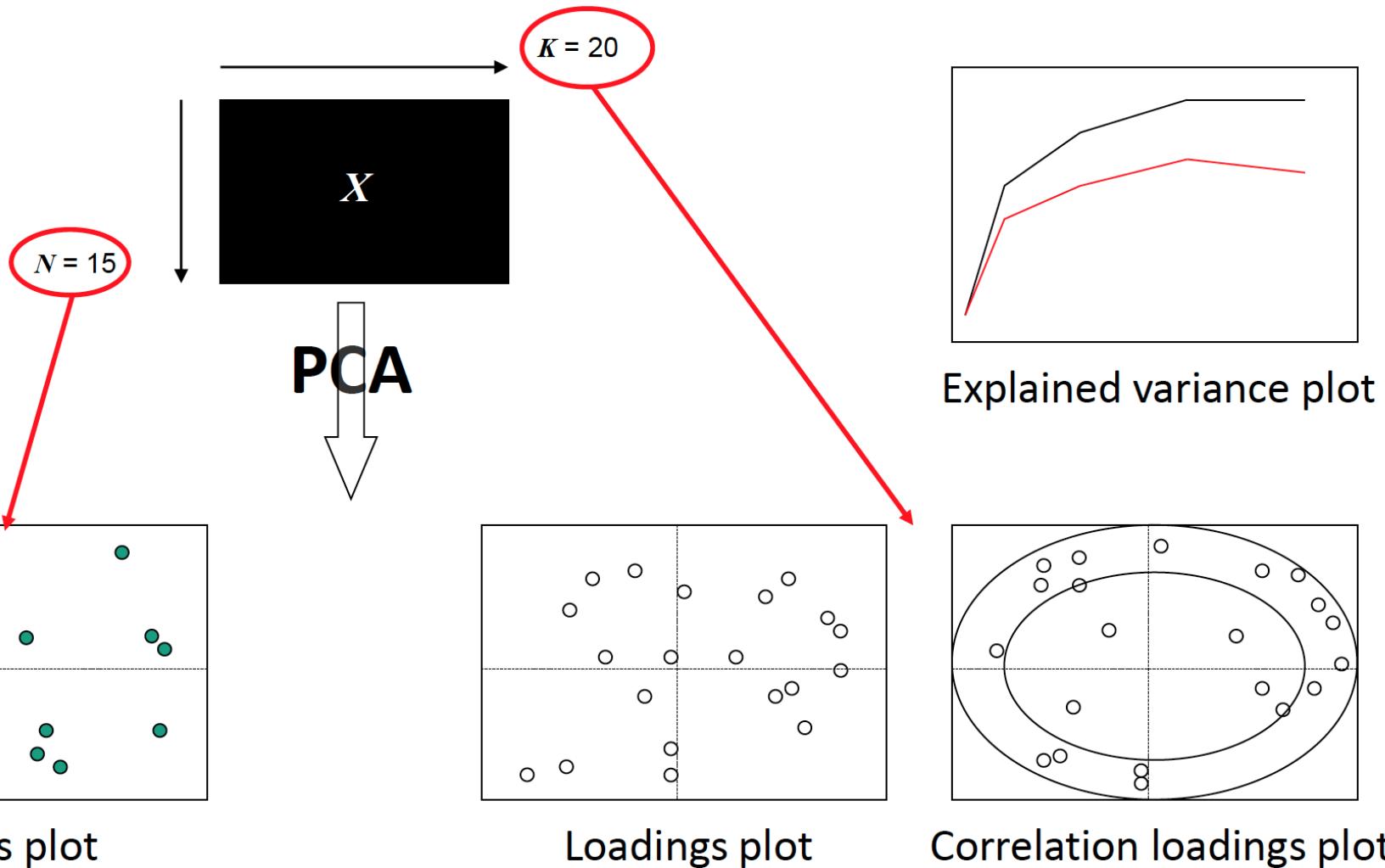
E : residuals / noise

Principal Components (PC's) describing the systematic variation in the data

PCA basics

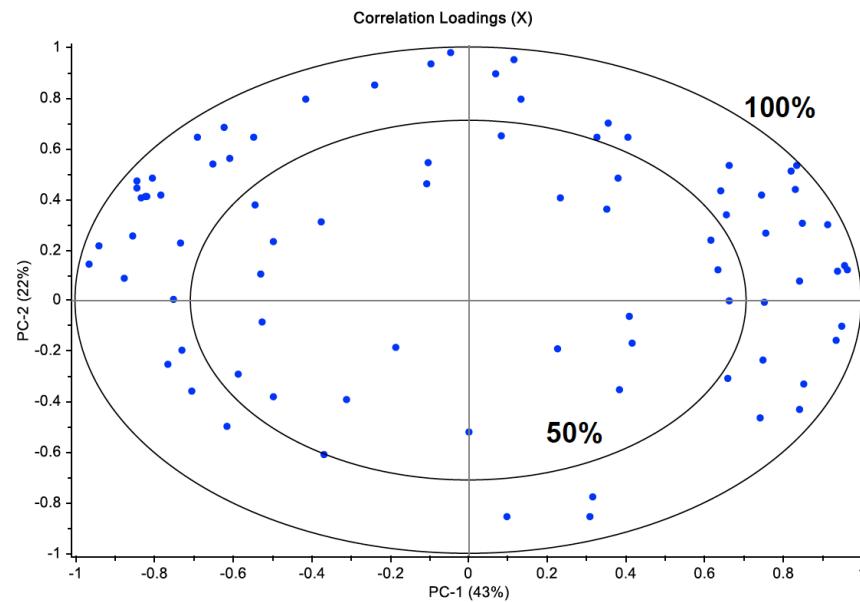
Data in this illustration consists of 15 observations and 20 variables → data matrix X of dimension (15×20)

U
B
M
N



PCA- correlation loadings

- Circles in the plot corresponding to various degrees of explained variances
- Typically one will present a circle for **100%** explained and for **50%** explained variance by the **two** components



PCA basics – centring and standardisation



- Only purpose of PCA is to look for directions with **high variance**
- This implies: if there are variables x_k in X that have a **larger variance** than others ...
 - they will be given **most** attention
 - → They will **dominate** the extracted components
 - → They will **dominate** the plots
- Generally one is interested in letting all variables play a role in the estimation of components (there are exceptions) → standardise variables x_k in X
- Matrices in multivariate statistics are always **either** centered or standardised

PCA basics – centring and standardisation



$$X = \begin{pmatrix} x_{11} & \cdots & x_{1K} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NK} \end{pmatrix}$$

- Number of **objects (rows)**:
 - $n = 1 \dots N$
- Number of **variables (columns)**:
 - $k = 1 \dots K$
- Observed value x_{nk} for
 - n 'th object
 - k 'th variable

center

$$x_{nk,cent} = x_{nk} - \bar{x}_k$$

$$\bar{x}_k = \frac{1}{N} \sum_{n=1}^N x_{nk}$$

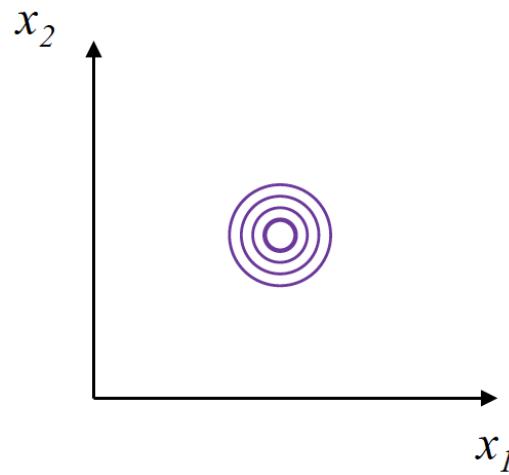
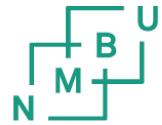
where

standardise

$$x_{nk,stand} = \frac{x_{nk} - \bar{x}_k}{\sigma_k}$$

$$\sigma_k = \sqrt{\frac{1}{N} \sum_{n=1}^N (x_{nk} - \bar{x}_k)^2}$$

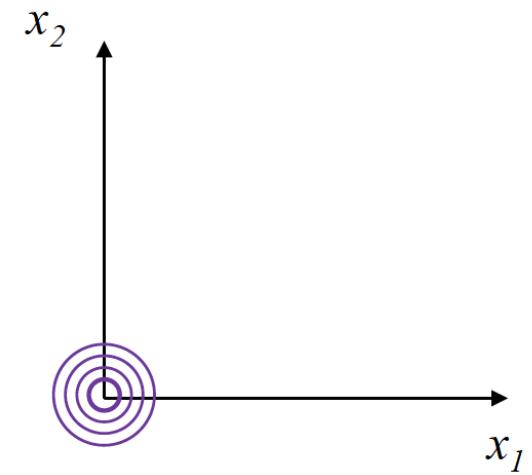
PCA basics – centring and standardisation



center data

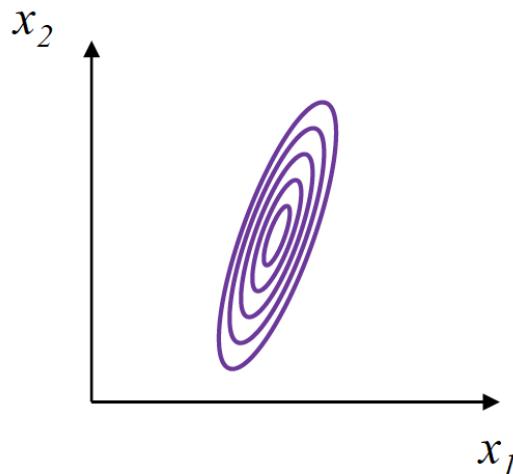


$$x_{nk,cent} = x_{nk} - \bar{x}_k$$

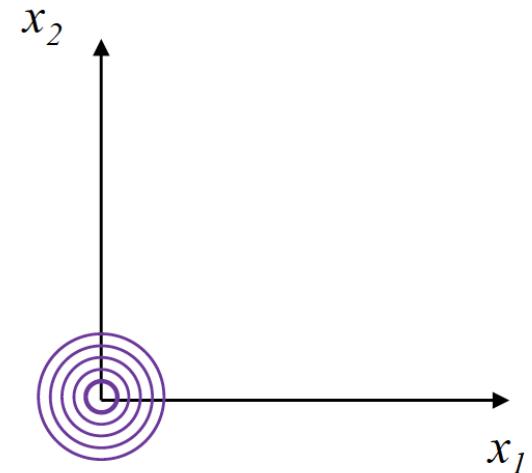
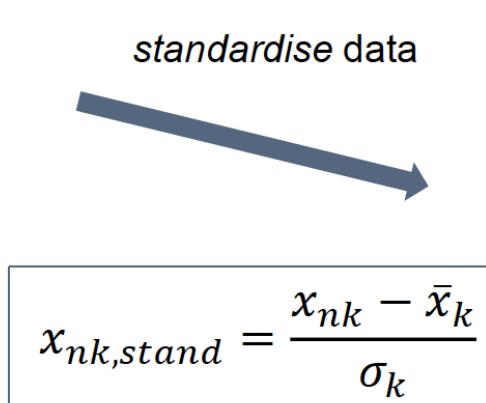
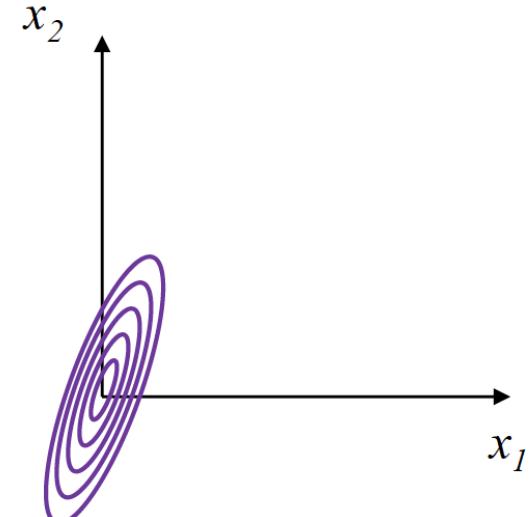
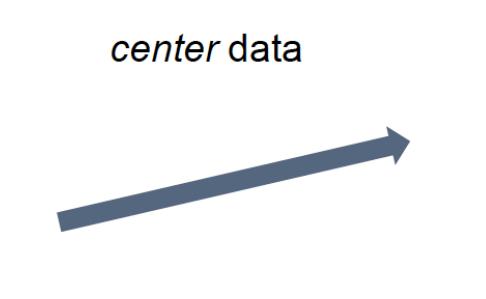


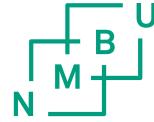
Equal variance of
 x_1 and x_2

PCA basics – centring and standardisation



Variance of x_2 is
larger than variance
of x_1

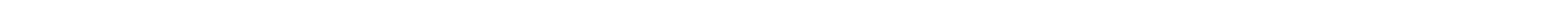




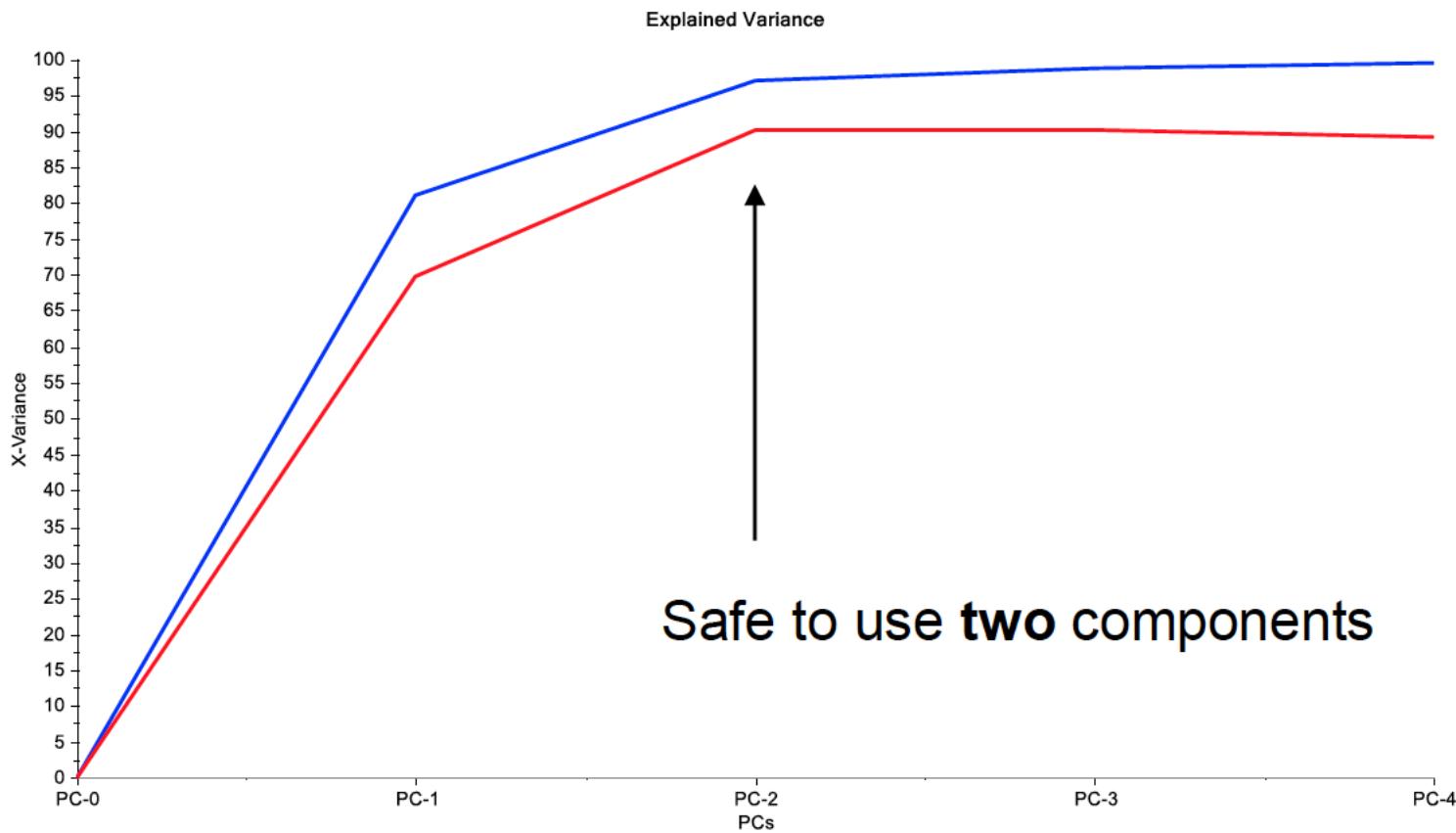
PCA - validation

Validation is necessary to gain knowledge on **how many** components are **appropriate** for the model, i.e. how many components can be used for interpretation and further analysis.

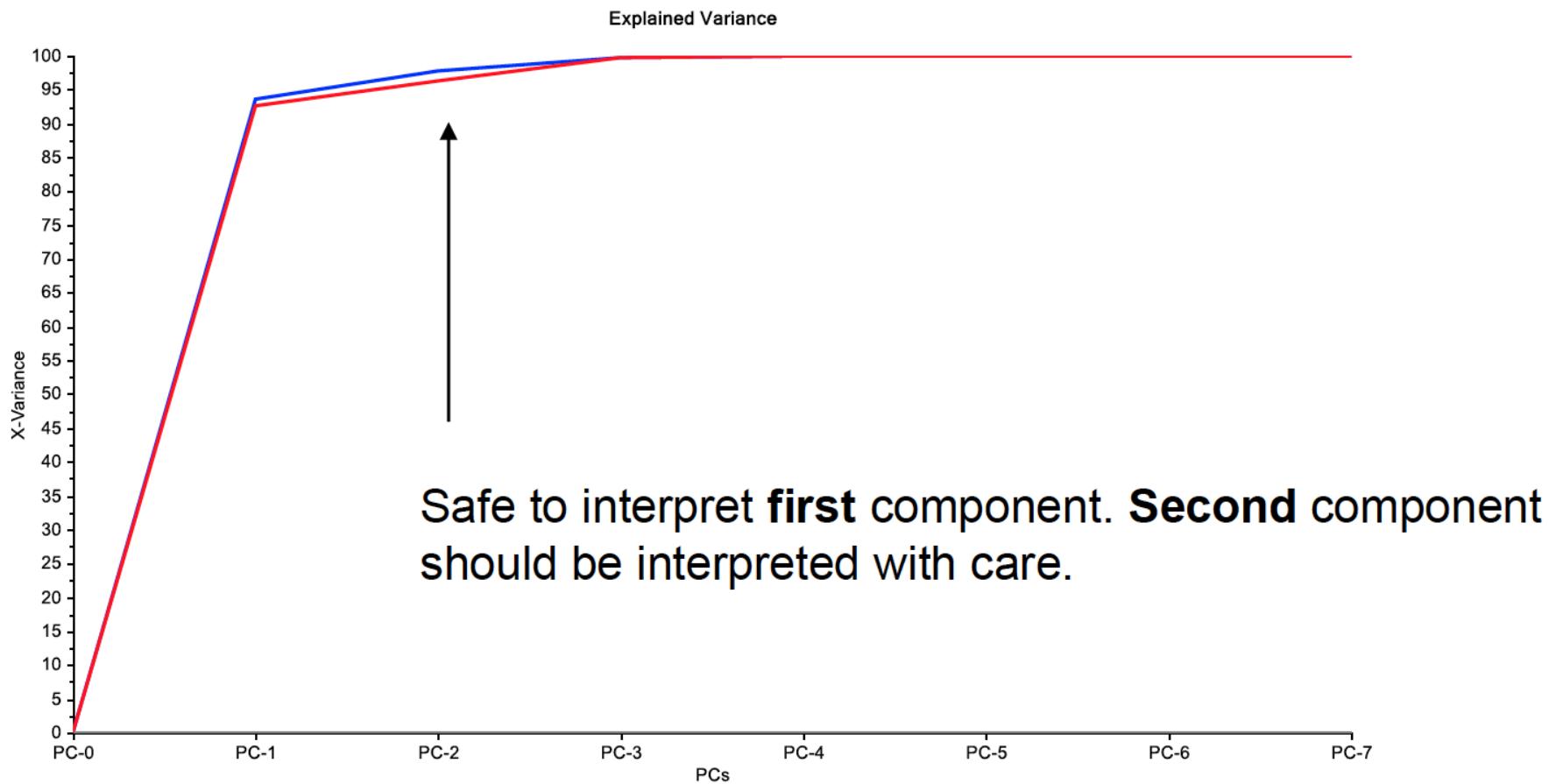
- Use of *internal cross validation* in PCA
- **K-fold** cross validation (number of folds / splits used)
- **LOO** cross validation (“Leave-one-out”)



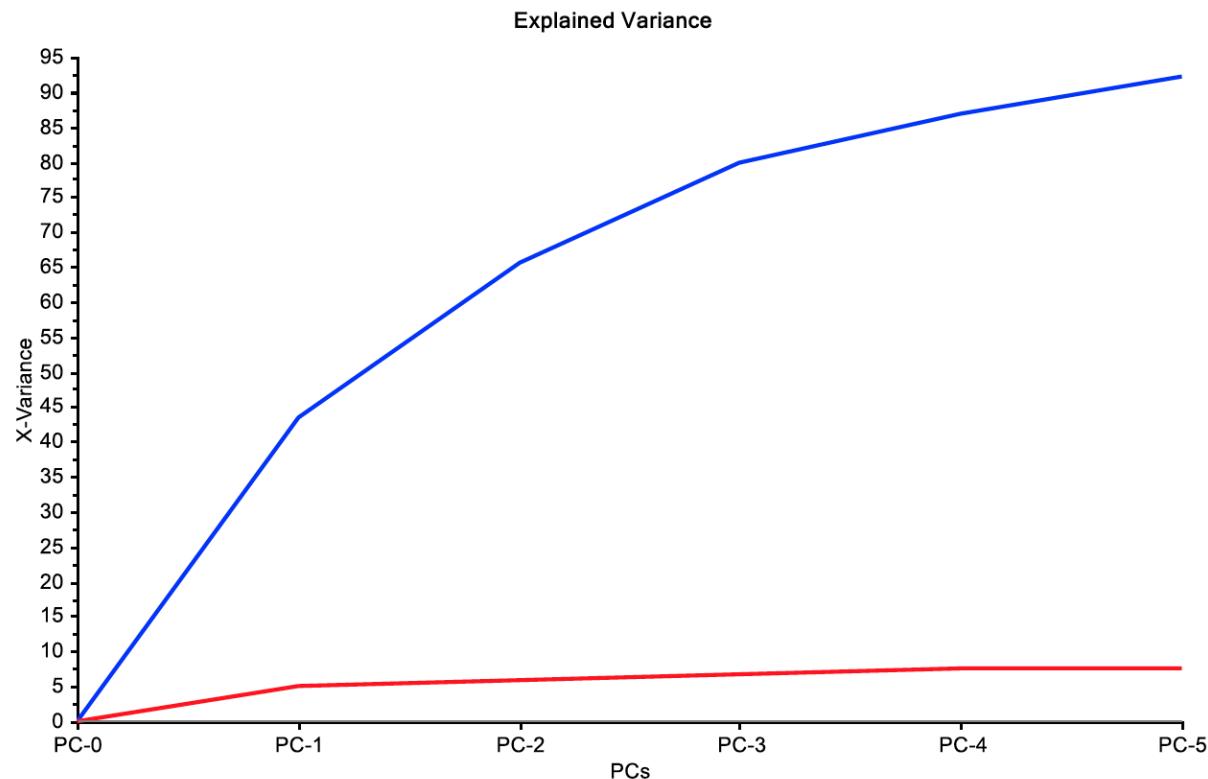
PCA - validation



PCA - validation



PCA - validation

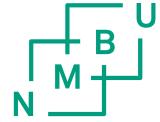


Poor model – may be a result of few objects that are very different from each other or overfitting

Hoggorm, HoggormPlot and examples

- **Hoggorm** package for multivariate statistics
 - GitHub: <https://github.com/olivertomic/hoggorm>
 - Read the Docs: <http://hoggorm.readthedocs.io/en/latest/>
- **HoggormPlot** package for convenient plotting of Hoggorm results
 - GitHub: <https://github.com/olivertomic/hoggormPlot>
 - Read the Docs: <http://hoggormplot.readthedocs.io/en/latest/>
- **Examples** of how to use Hoggorm illustrated in Jupyter notebooks
 - GitHub: <https://github.com/khliland/hoggormExamples>

Hoggorm



```
import hoggorm as ho
import hoggormplot as hopl
from sklearn import datasets

# import Iris data set

iris = datasets.load_iris()
X = iris.data
Y = iris.target

# Get the variables
iris_varNames = list(iris.feature_names)

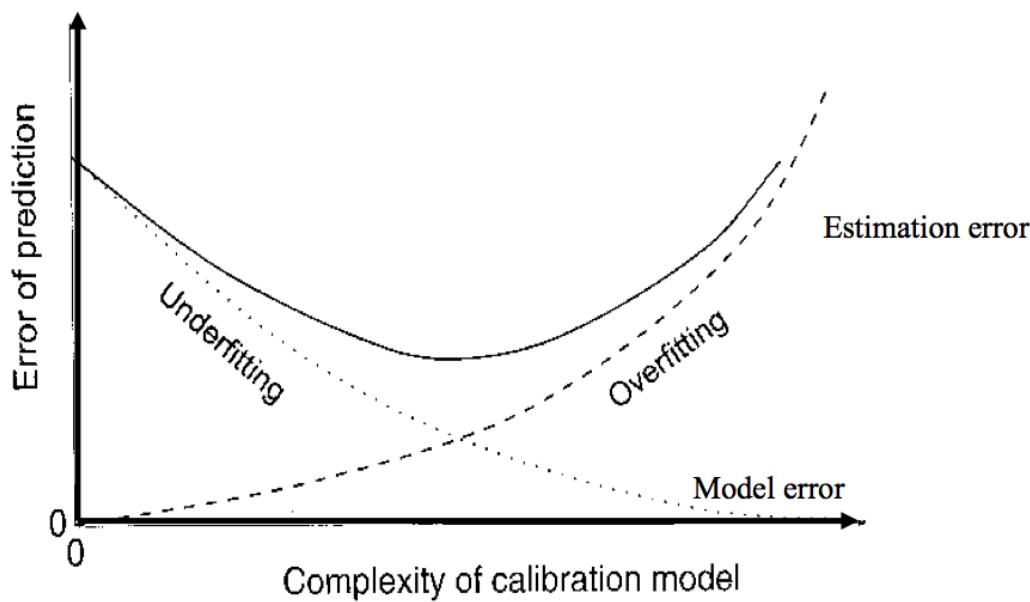
# Get the objects
iris_objNames = list(iris.target_names)

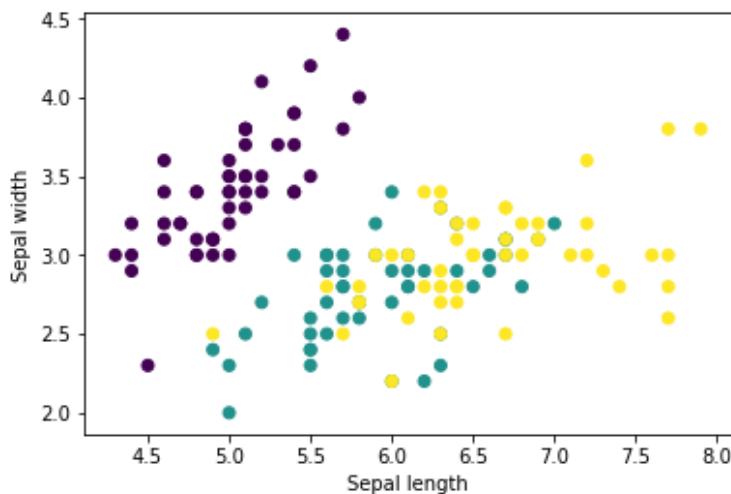
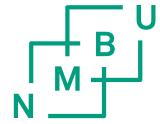
model_01 = ho.nipalsPCA(arrX=X, Xstand=True, cvType=["loo"], numComp=3)

hopl.plot(model_01, plots=[1, 2, 3, 6], line=True)
hopl.plot(model_01)
hopl.plot(model_01, plots=['scores', 'loadings', 'explainedVariance'])
```

HOW MANY COMPONENTS?

- «Use the number of components that do not produce overfitting of model»
- Explained variance for PCA
- Prediction error for PCR/PLS





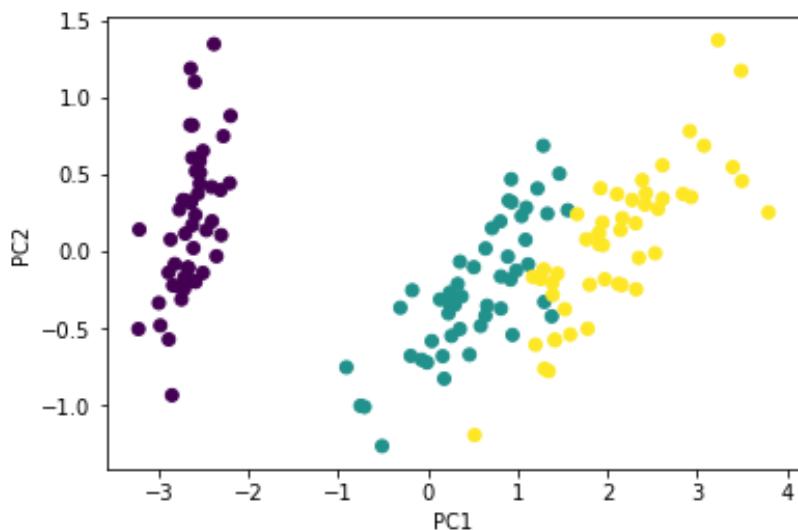
```

import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA

# import the Iris data
iris = datasets.load_iris()
X = iris.data
y = iris.target

plt.figure()
plt.scatter(X[:,0],X[:,1],c=y)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()

```



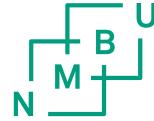
```

#simple PCA

X_reduced = PCA(n_components=3).fit_transform(iris.data)
plt.figure()
plt.scatter(X_reduced[:,1],X_reduced[:,2],c=y)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()

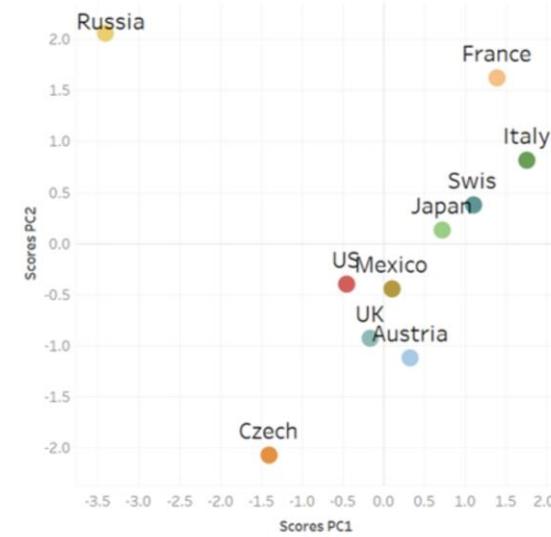
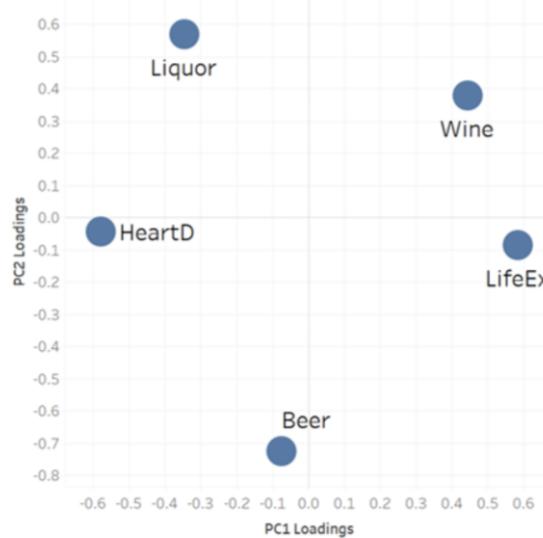
```

Causality. ?

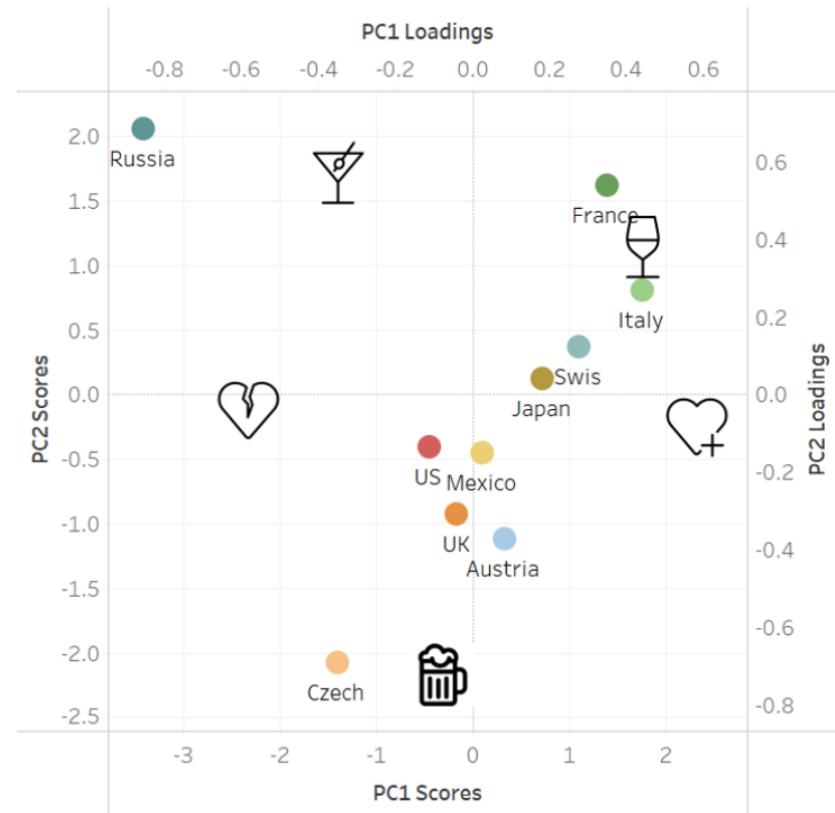
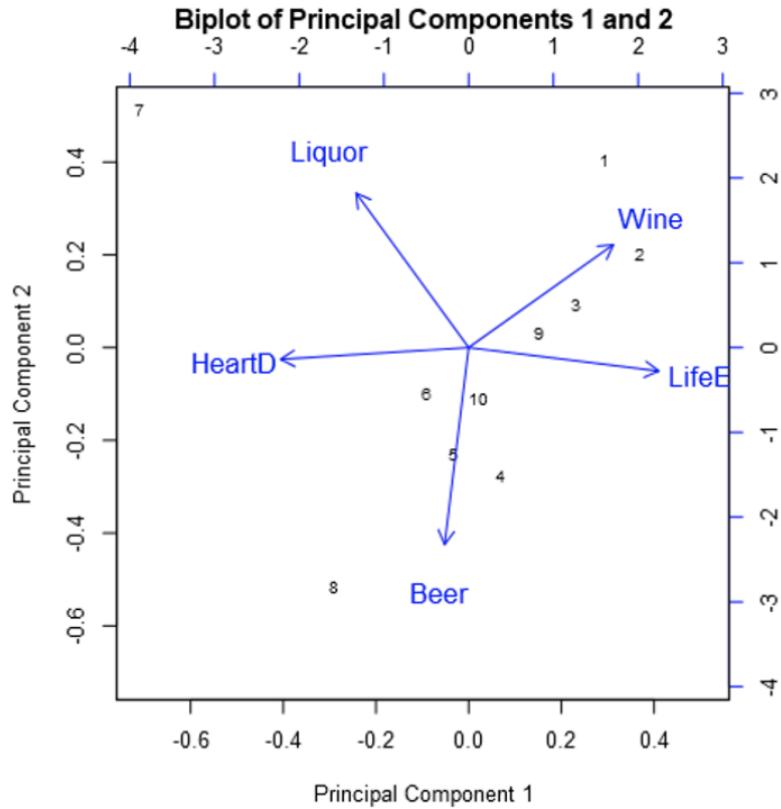


Data set from Time Magazines 1996

| | Liquor | Wine | Beer | Life Ex | Heart D |
|---------|--------|------|-------|---------|---------|
| France | 2.5 | 63.5 | 40.1 | 78.0 | 61.1 |
| Italy | 0.9 | 58.0 | 25.1 | 78.0 | 94.1 |
| Swis | 1.7 | 46.0 | 65.0 | 78.0 | 106.4 |
| Austria | 1.2 | 15.7 | 102.1 | 78.0 | 173.0 |
| UK | 1.5 | 12.2 | 100.0 | 77.0 | 199.7 |
| US | 2.0 | 8.9 | 87.8 | 76.0 | 176.0 |
| Russia | 3.8 | 2.7 | 17.1 | 69.0 | 373.6 |
| Czech | 1.0 | 1.7 | 140.0 | 73.0 | 283.7 |
| Japan | 2.1 | 1.0 | 55.0 | 79.0 | 34.7 |
| Mexico | 0.8 | 0.2 | 50.4 | 73.0 | 36.4 |



<https://www.thedataschool.co.uk/robbin-vernooij/principal-component-analysis-alteryx-example-pinguin/>



- Does this mean that you live longer if you drink wine, shorter if you drink Liquor?
- NO
- The analysis is about correlation, not causality



MULTIVARIATE ANALYSIS ON IMAGES

- Features in images are used in analysis
 - Matrix of Objects Shape, size, distribution etc
- Surface texture analysis
 - Image Surface texture



UNIVARIATE ANALYSIS ON IMAGES

- OBJECT analysis of chocolate

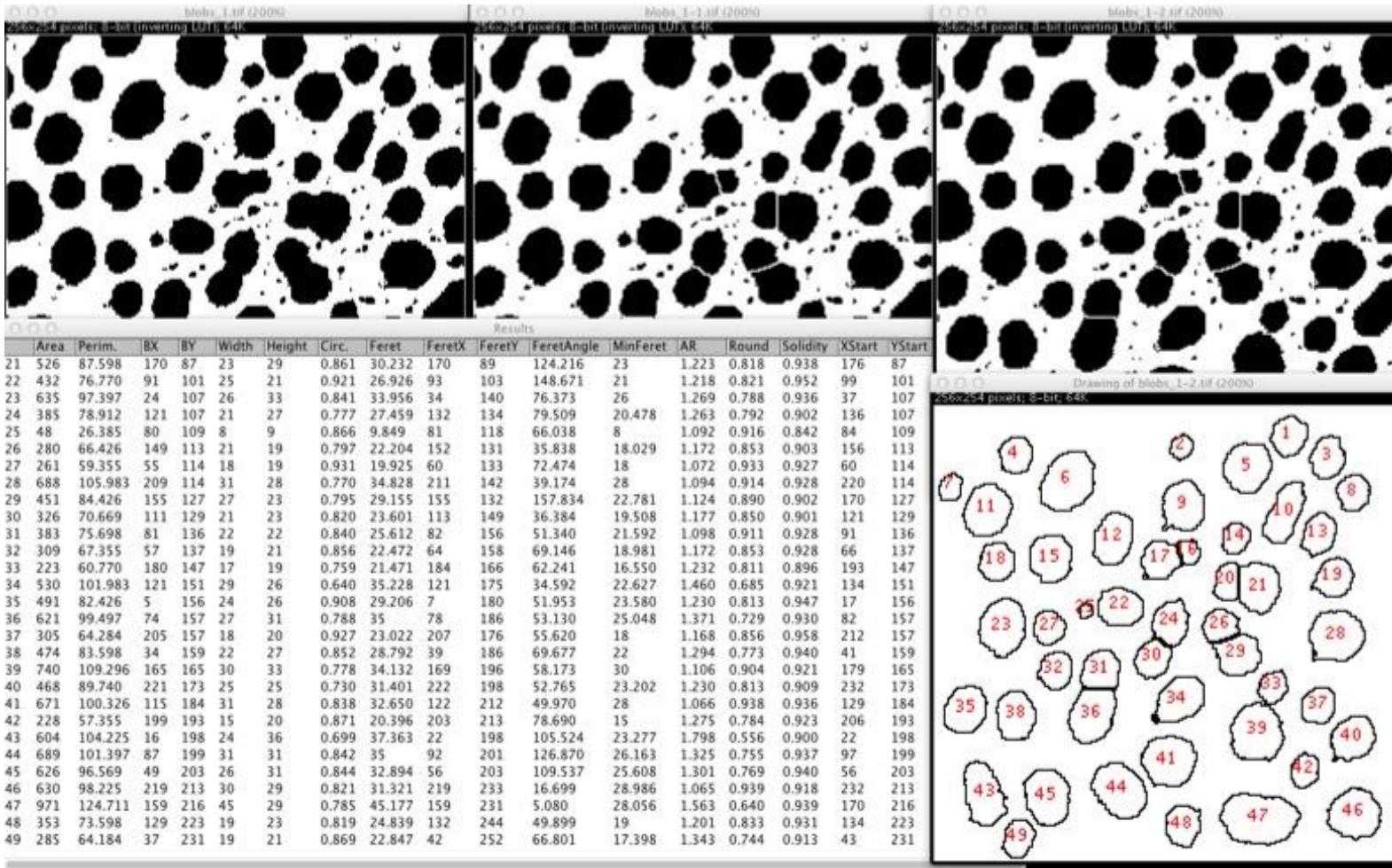


UNIVARIATE ANALYSIS ON IMAGES

- OBJECT analysis

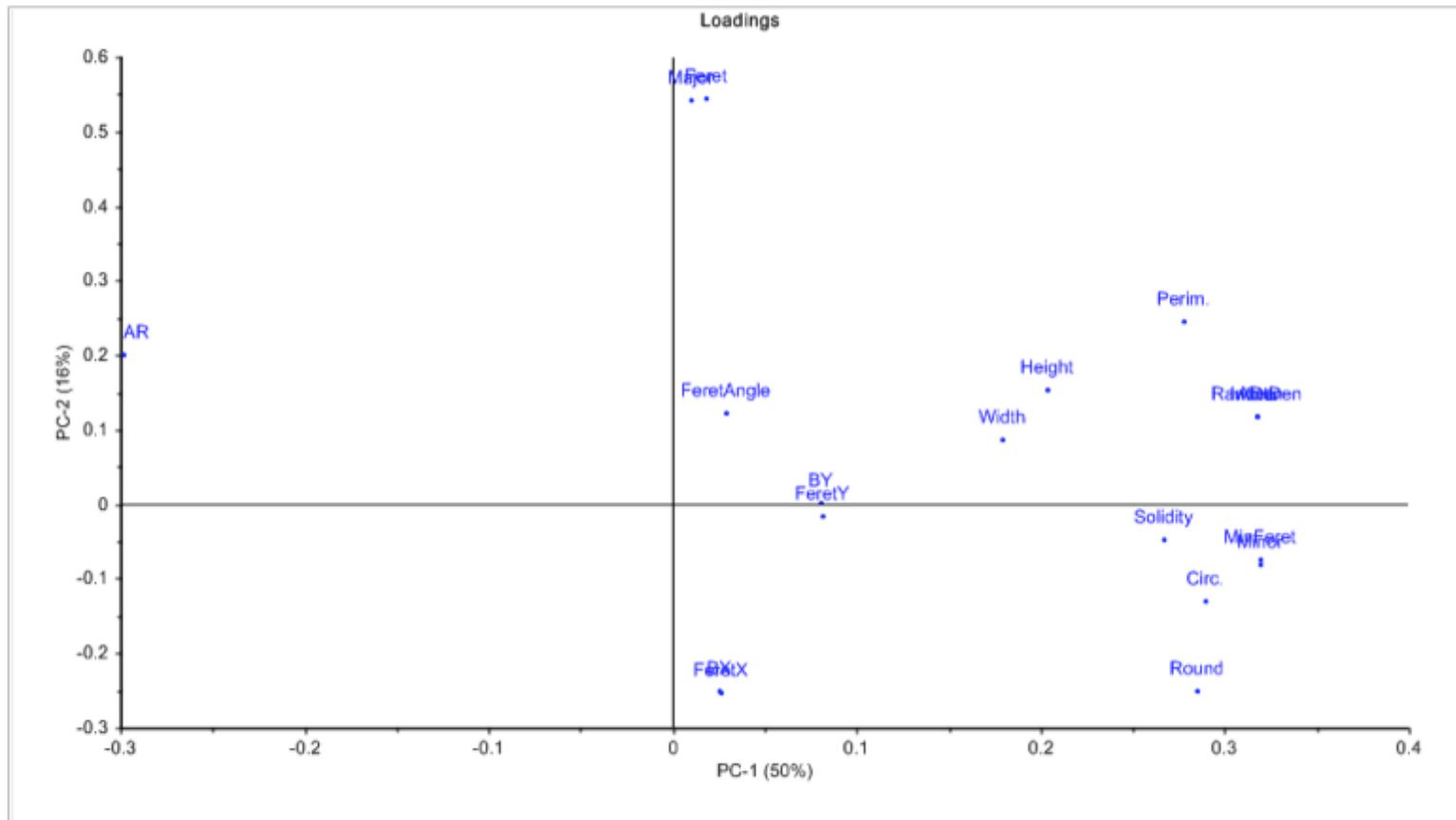


Figur 1. Sortert mhp «Roundness»



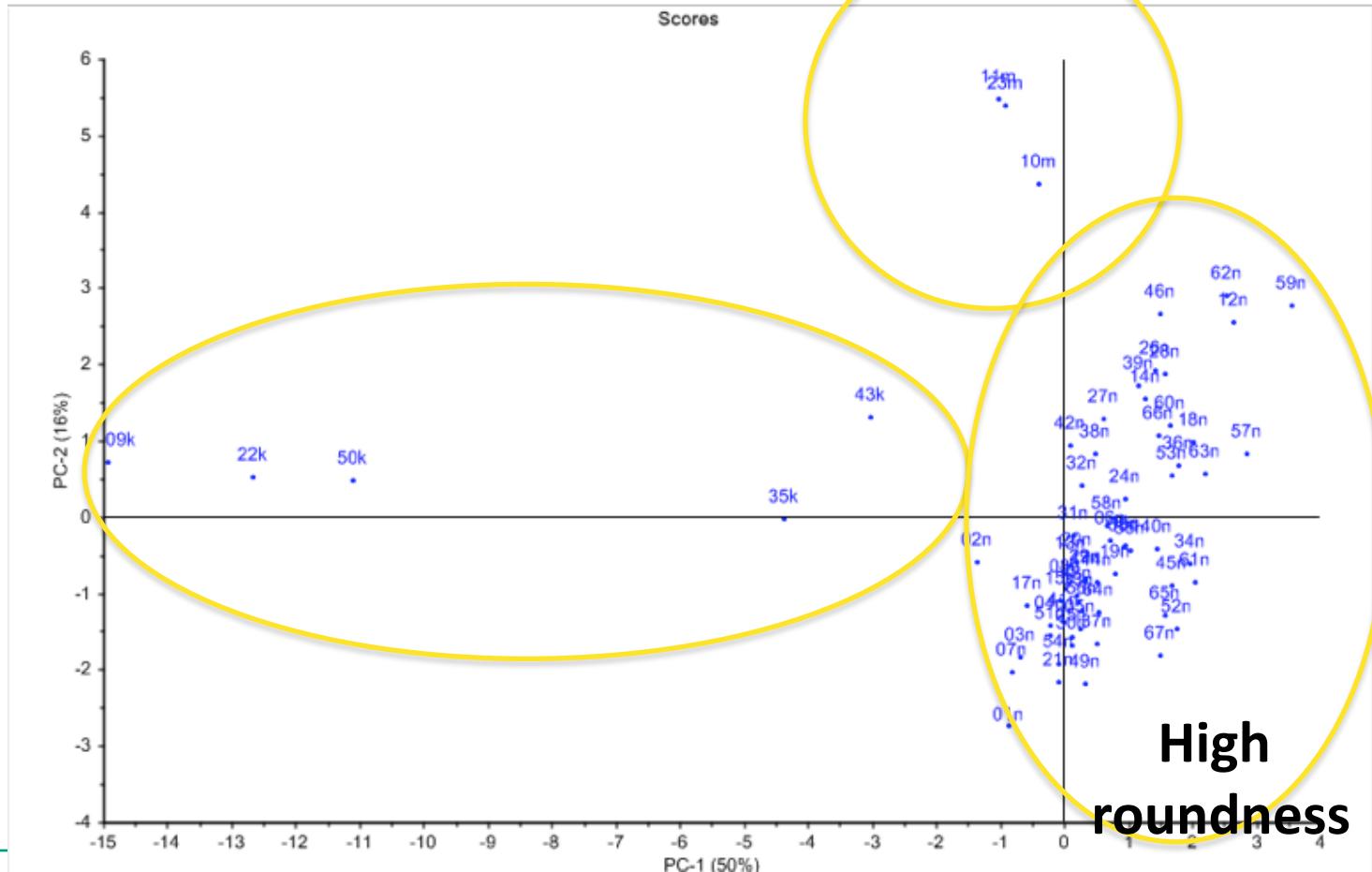
MULTIVARIATE ANALYSIS ON IMAGES

- PCA loadings on shape descriptors (area, roundness, ellipse etc)



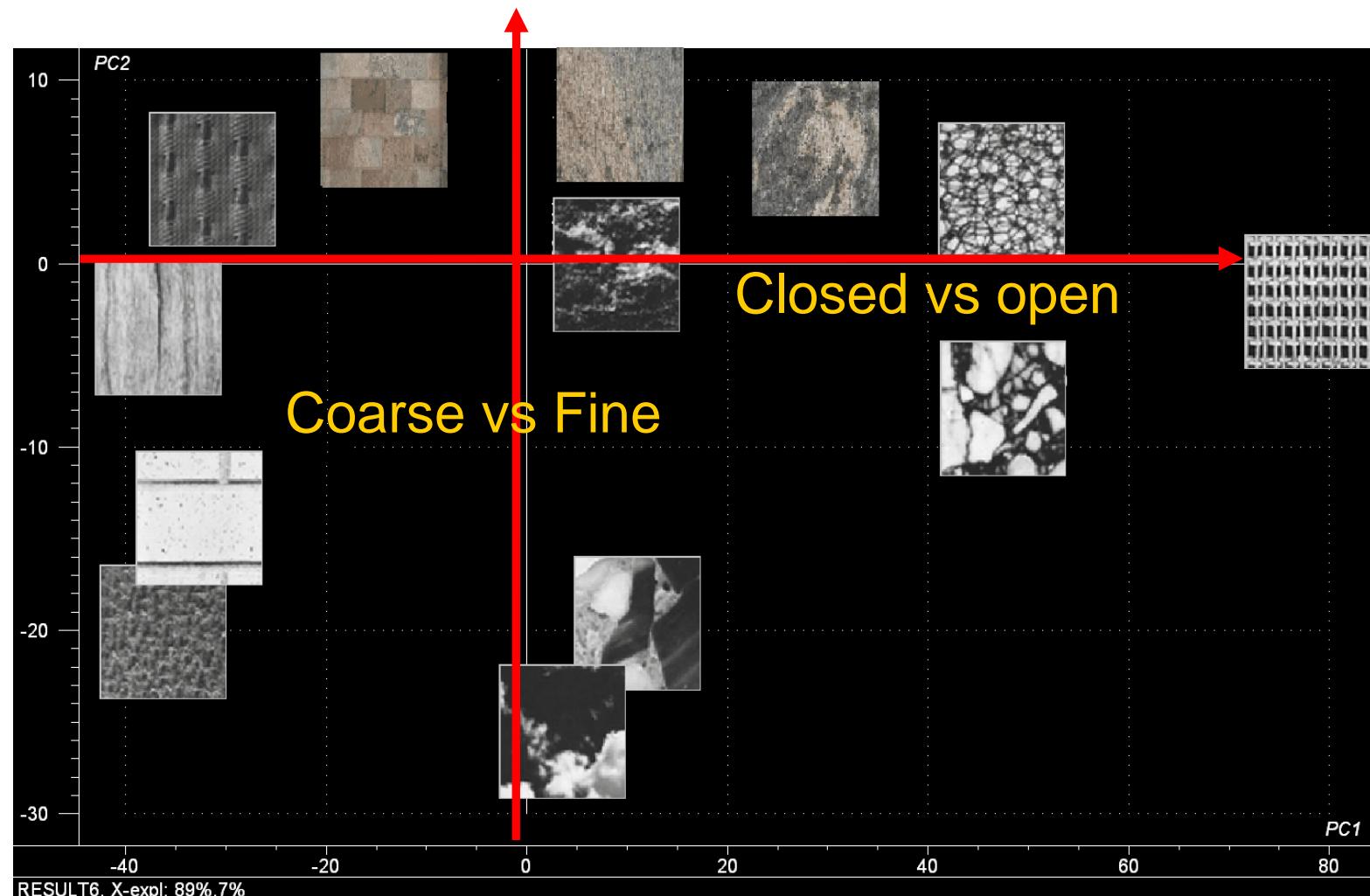
MULTIVARIATE ANALYSIS ON IMAGES

- PCA scores on shape descriptors (area, roundness etc)

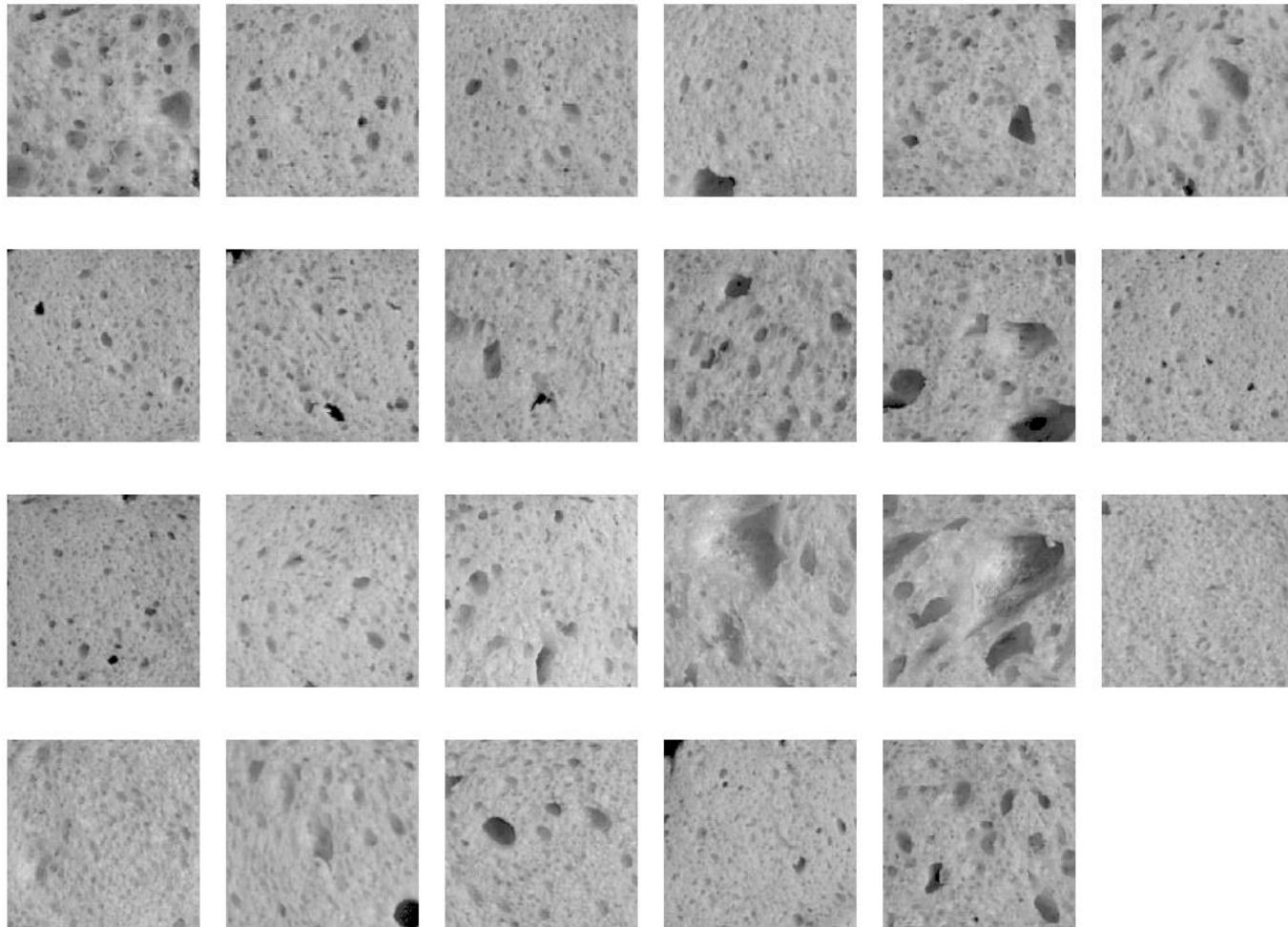
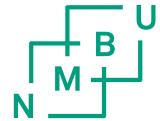


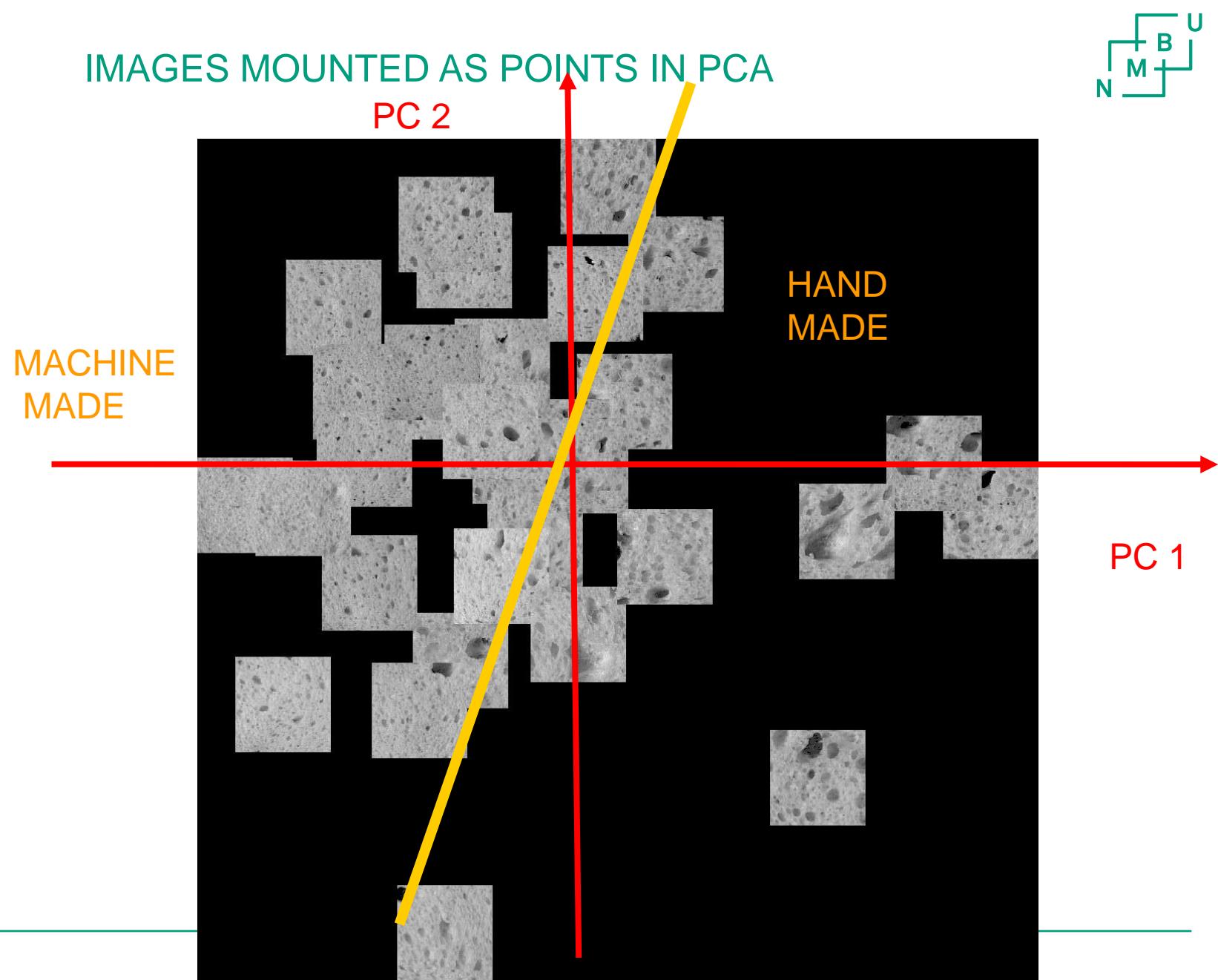
Principal Components Analysis of textures

U
B
M



Baguette textures and modeling of sensory porosity





Frost damage detection

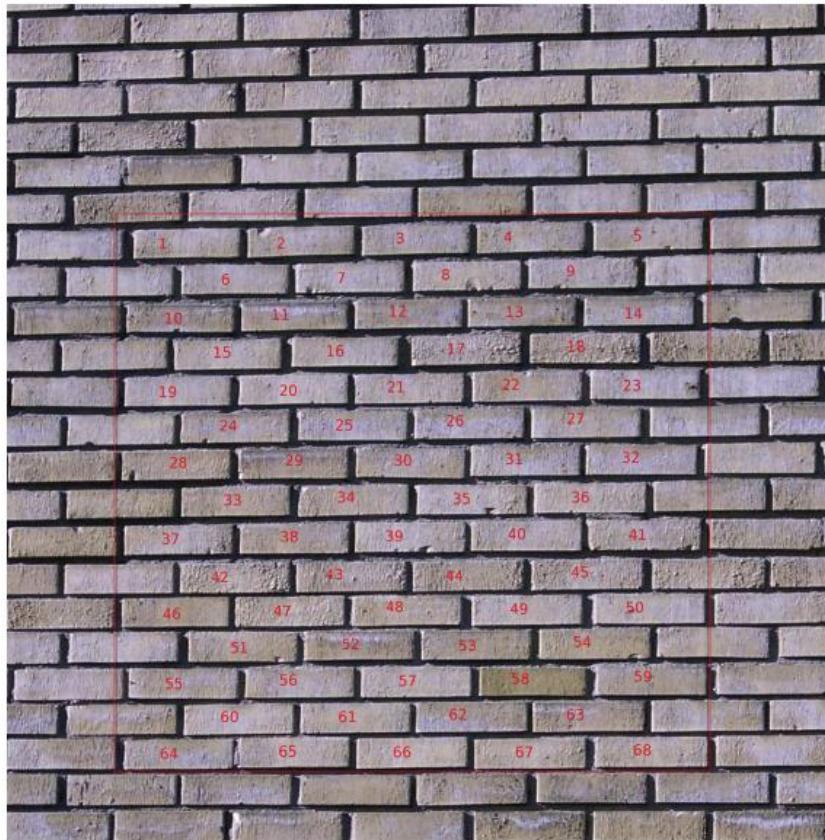
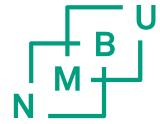


Fig 3: The original wall

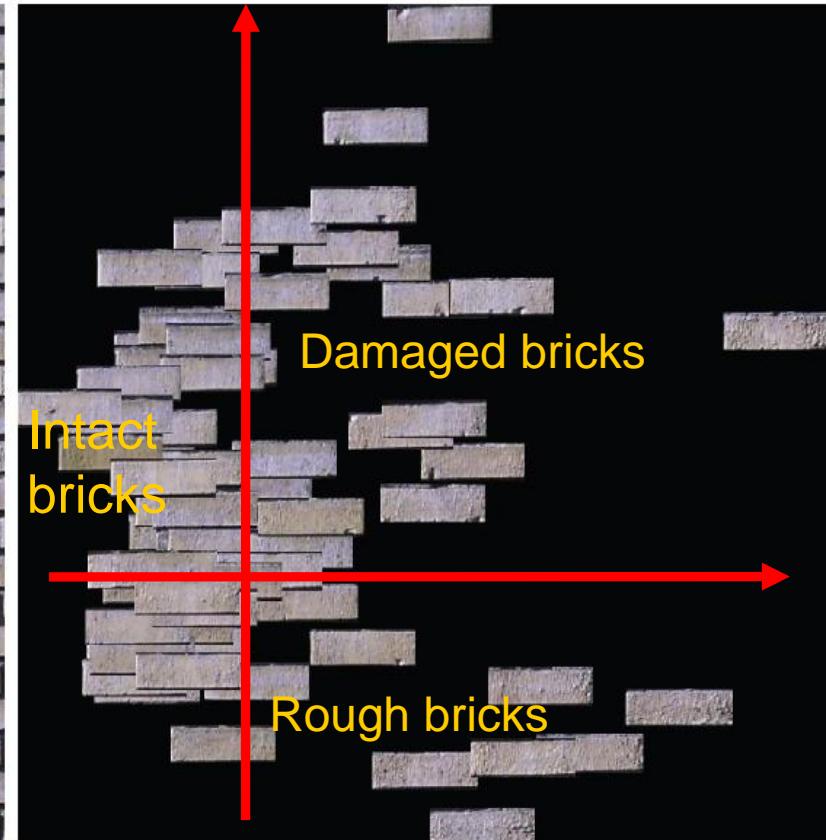
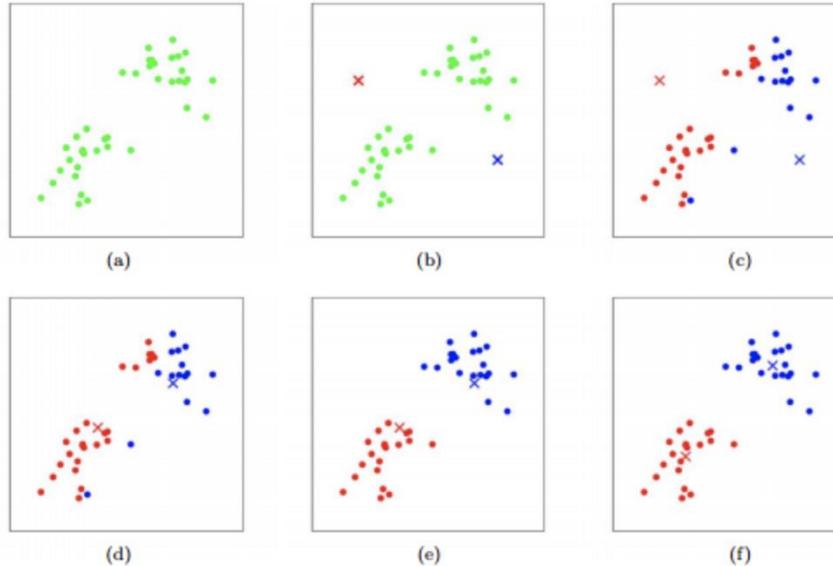


Fig 4: The bricks mounted in a PCA plot

Clustering

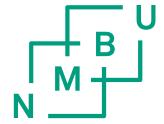


- K-means clustering
 - Separates samples into k groups of equal variance

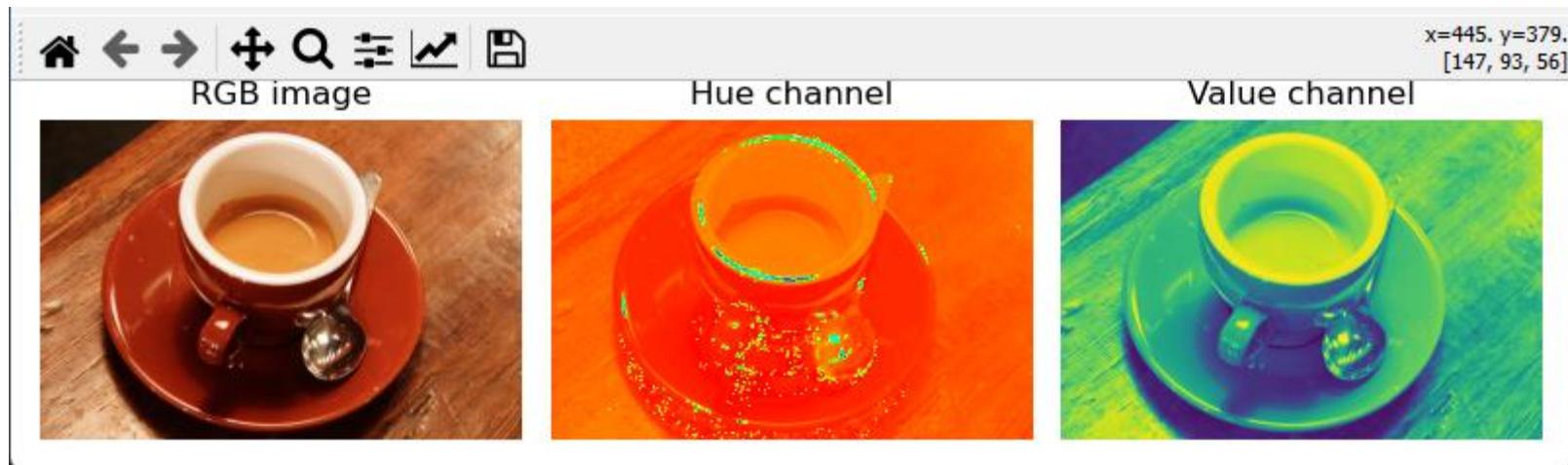


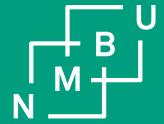
– https://www.youtube.com/watch?v=4b5d3muPQmA&ab_channel=StatQuestwithJoshStarmer

– <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>



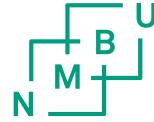
Color models in python





INF250

Spectral imaging

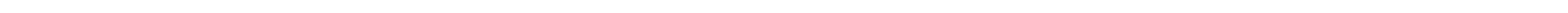


Learning goals

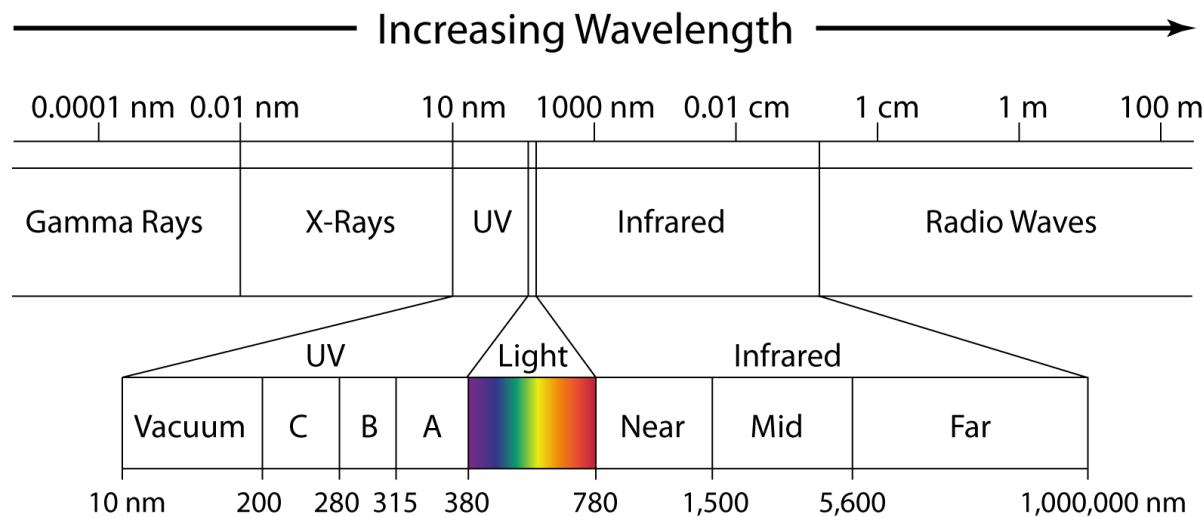
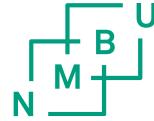
- What is multispectral and hyperspectral imaging ?
- What kind of cameras are used for hyperspectral imaging
- Give some examples of research one can do with hyperspectral imaging

Next lecture:

- Describe how PCA works on a hyperspectral image
- Describe how k-means clustering works
- Describe supervised classification



Spectral imaging



Conventional imaging

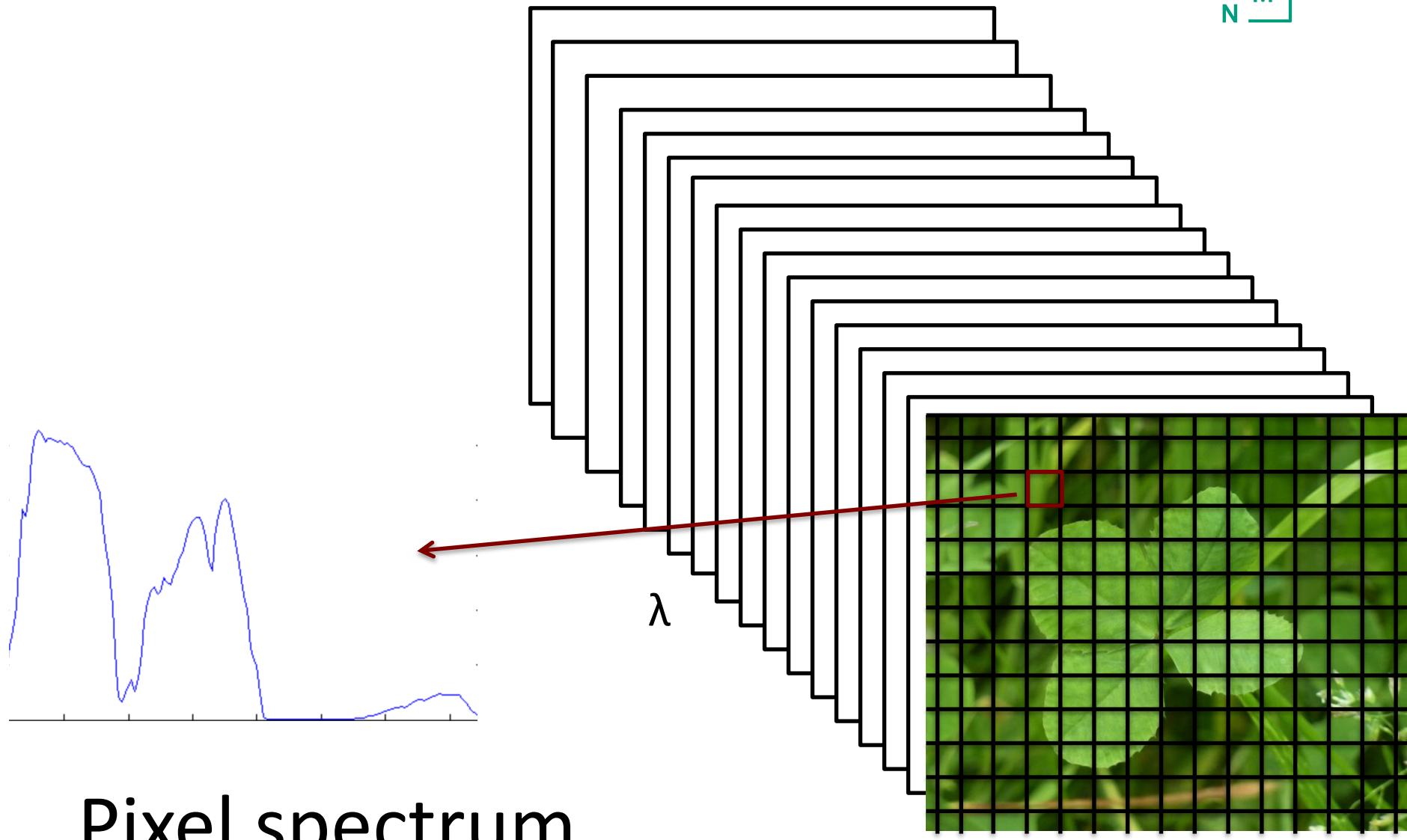


Multispectral imaging



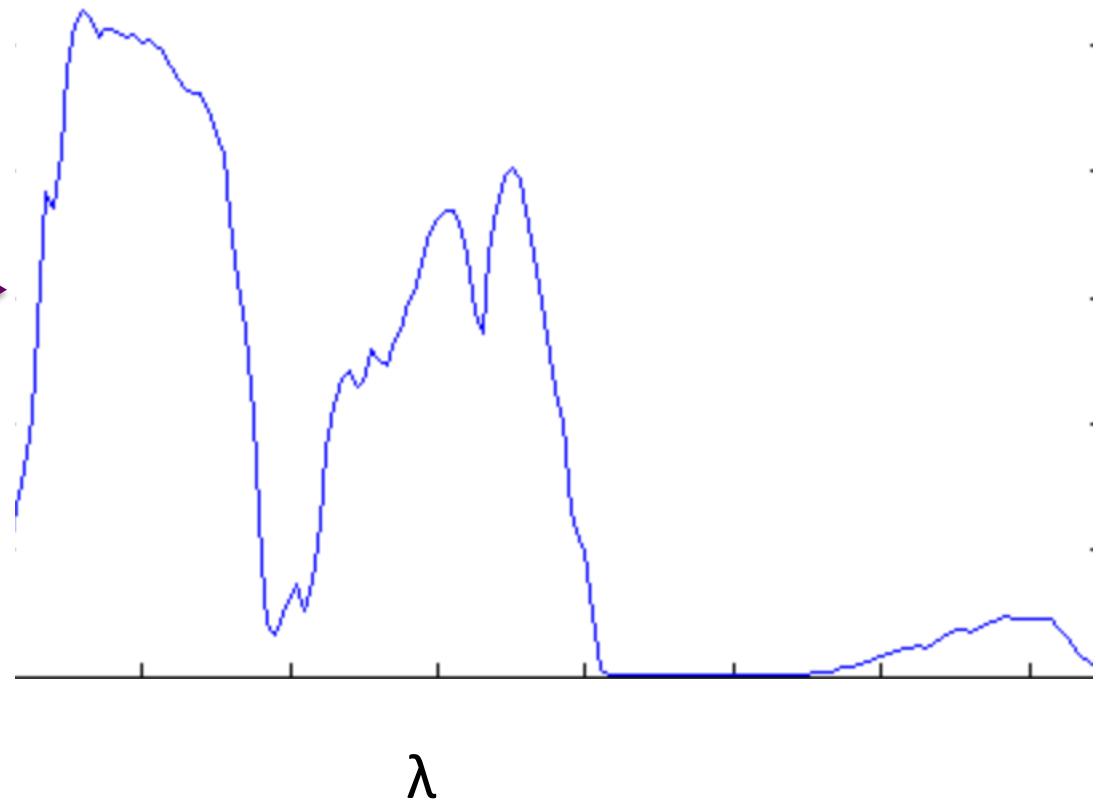
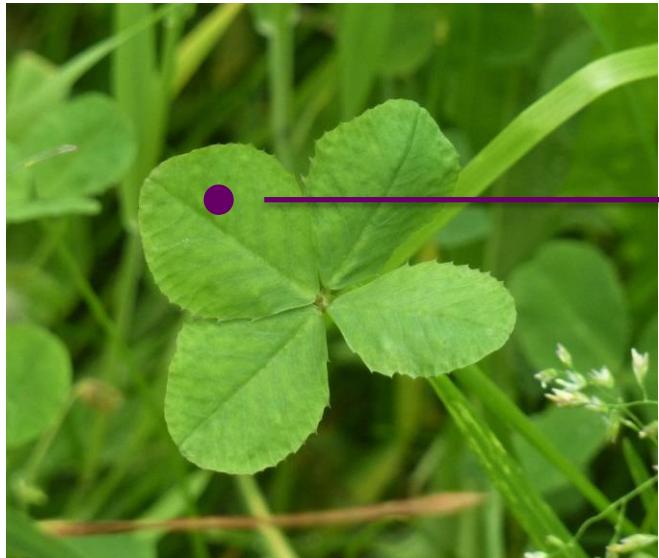
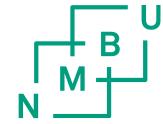
Hyperspectral imaging

N M B U

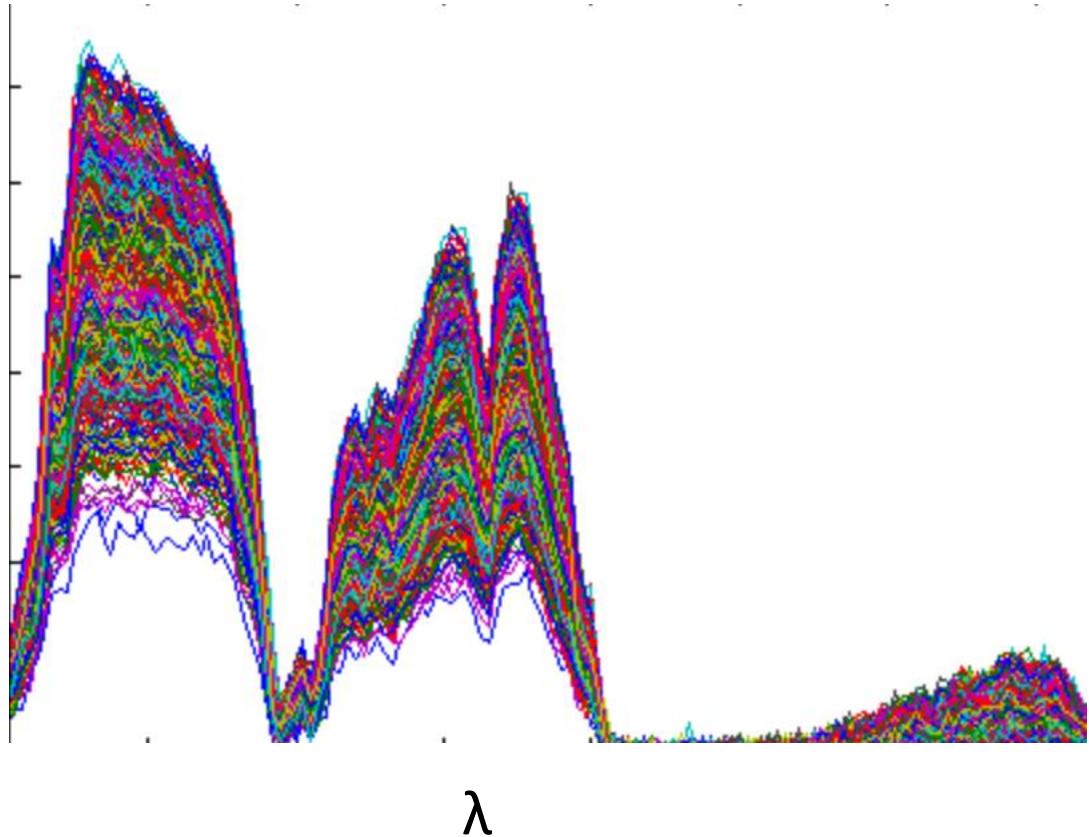
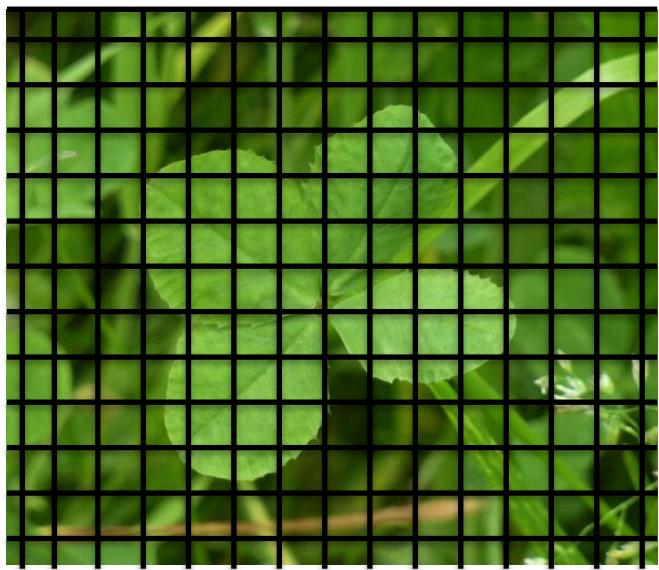
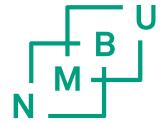


Pixel spectrum

Conventional spectroscopy



Hyperspectral

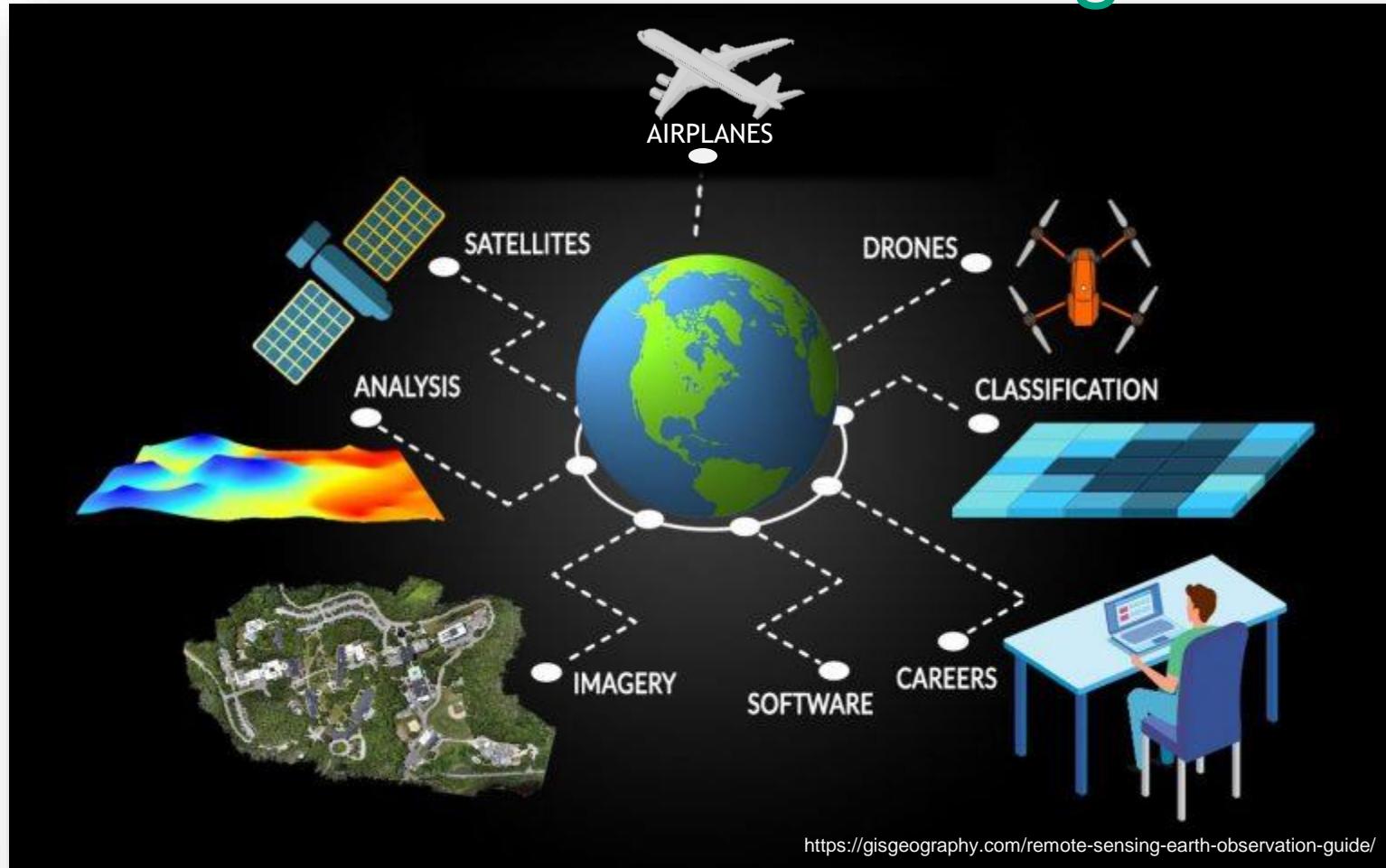


<http://www.markelowitz.com/Hyperspectral.html>

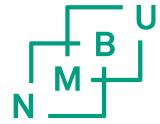
<http://www.microimages.com/documentation/Tutorials/hyprspec.pdf>

<https://www.youtube.com/watch?v=EaeRlzm-tWM>

Power of Remote Sensing

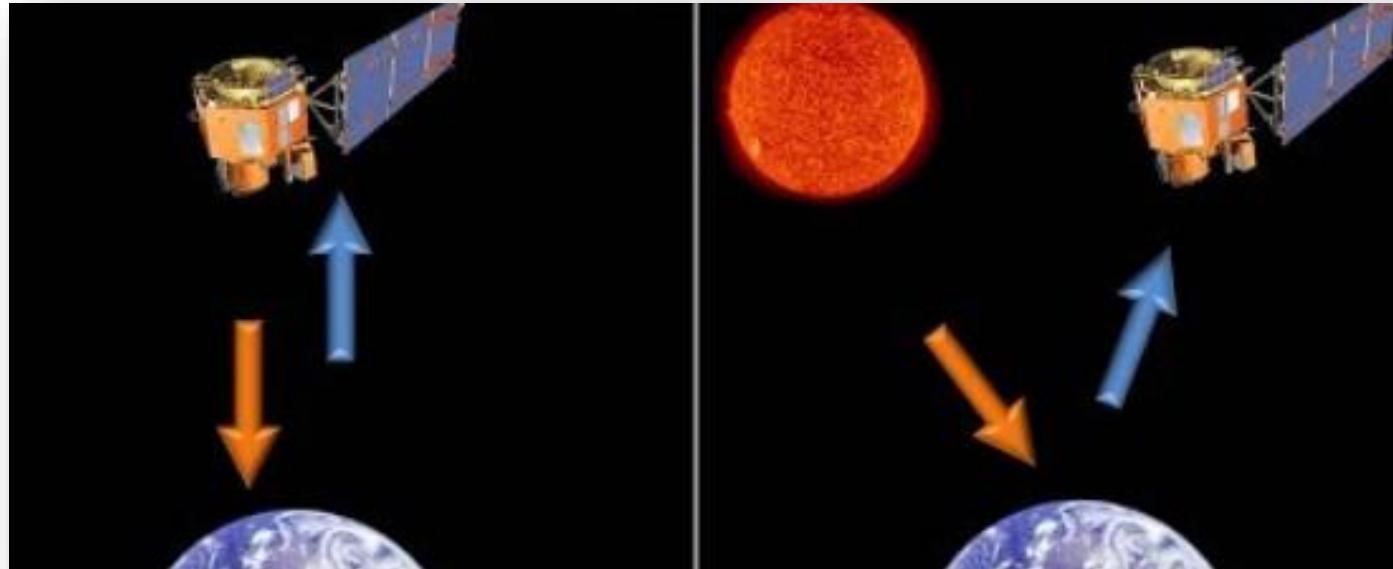


Active vs. Passive Remote Sensing



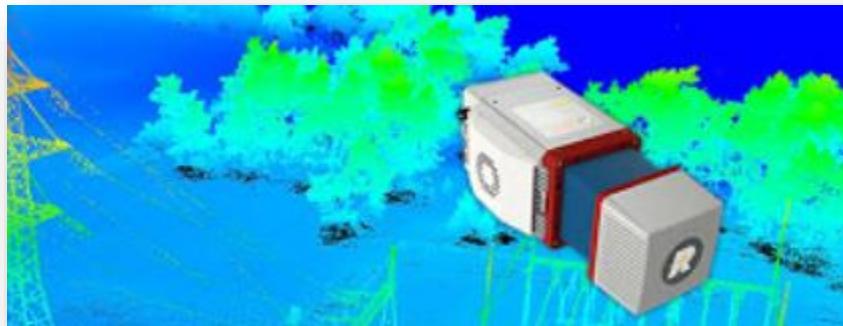
ACTIVE

PASSIVE



Active

- LiDAR (Light Detection and Ranging)

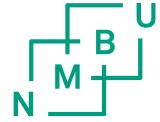


Passive

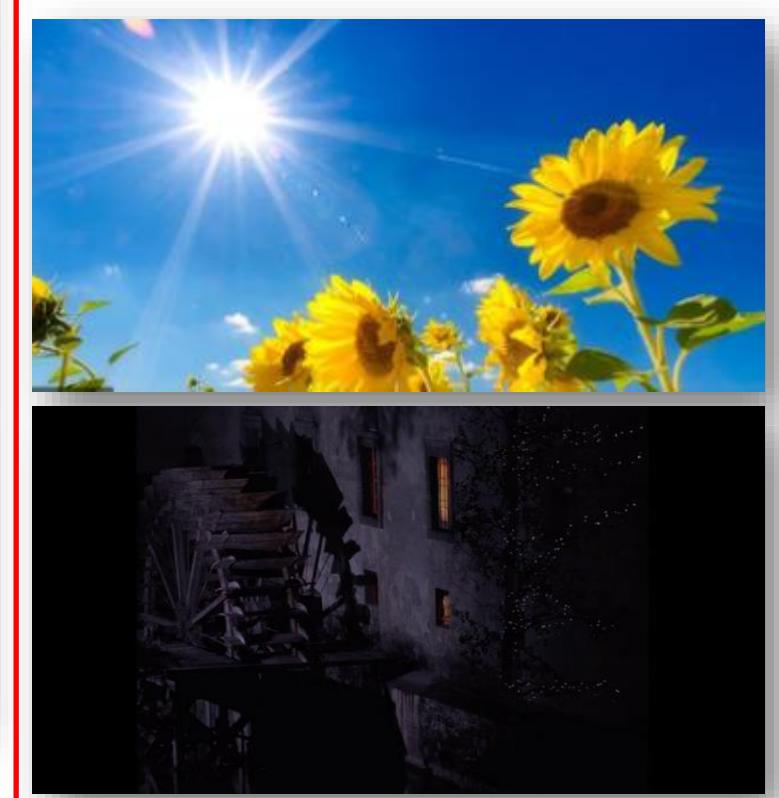
- Multi-/ hyperspectral sensor



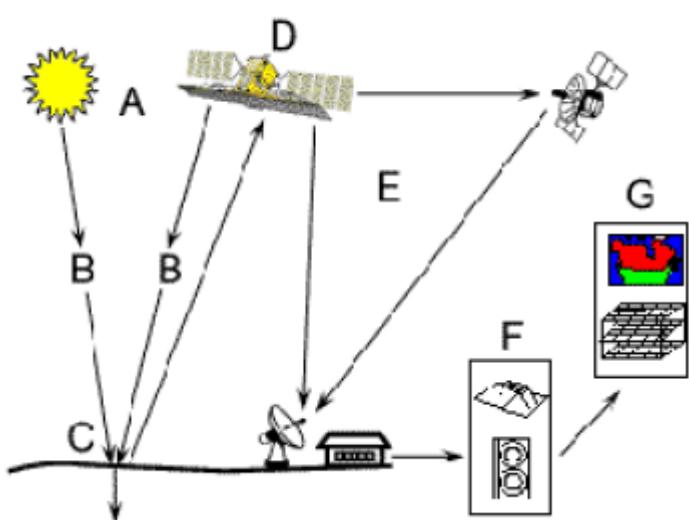
Is a camera an active or passive sensor?



Hint:



Passive remote sensing



A Energy source/Illumination

B Interaction with the Atmosphere

C Interaction with the Target

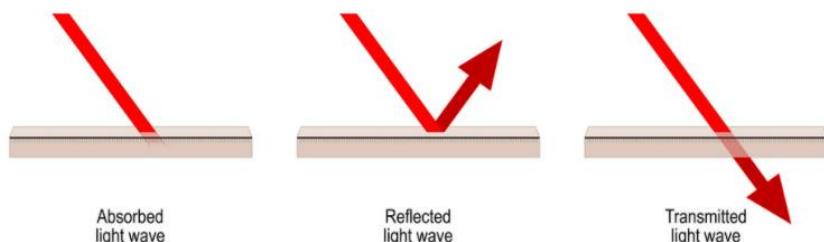
D Sensor is recording energy

E Transmission, Reception, Processing

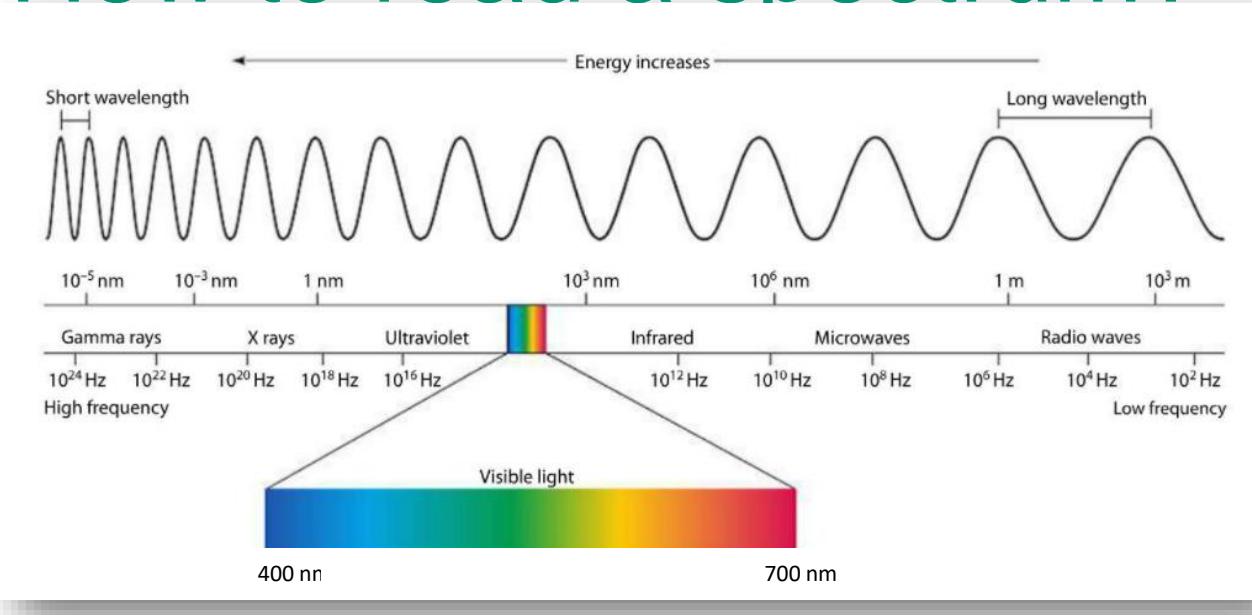
F Interpretation, Analysis

G Application

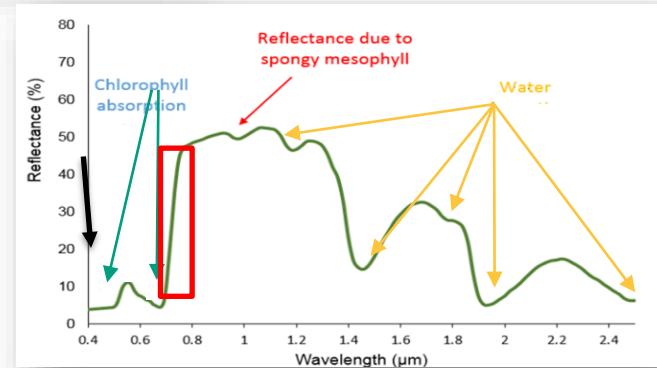
Light absorption, reflection, and transmission



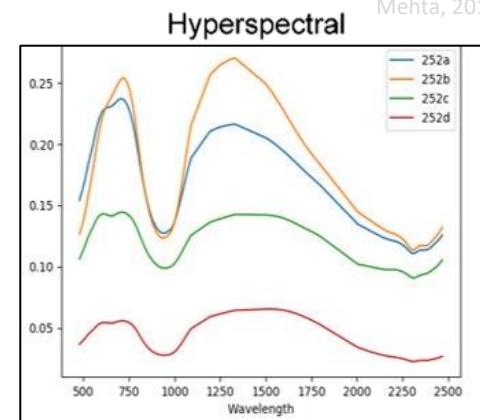
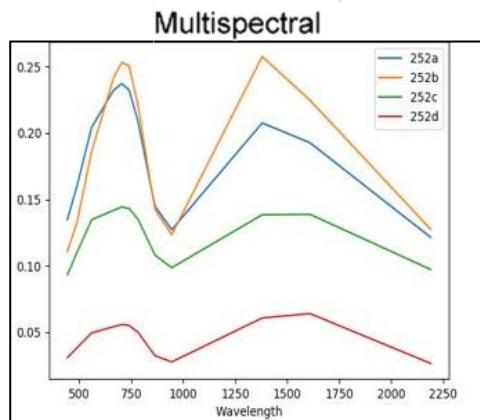
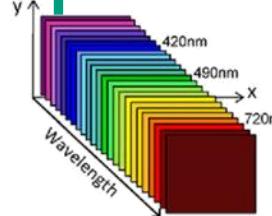
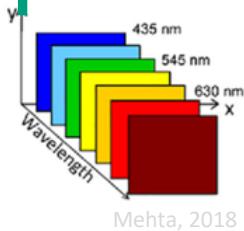
How to read a spectrum?



Red edge:
scattering due to leaf internal structure in the NIR region

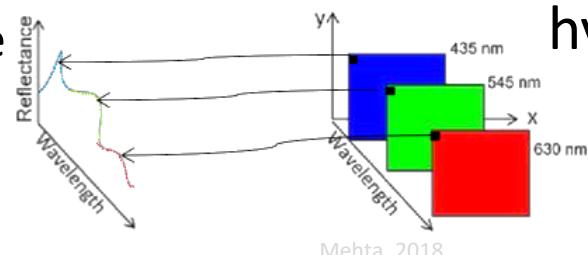


Multispectral vs. Hyperspectral



multi – more than one

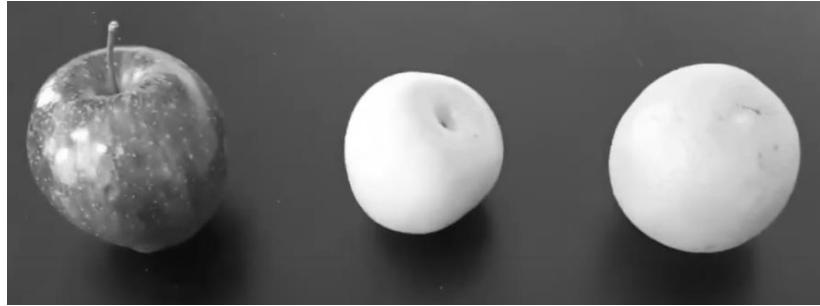
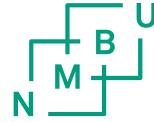
hyper – over, excessively



Spectroscopy

RGB

Image Analysis – Why hyperspectral?



- Colour is important for classification tasks
- BUT: even though we only see in RGB we aren't just limited to these three colours

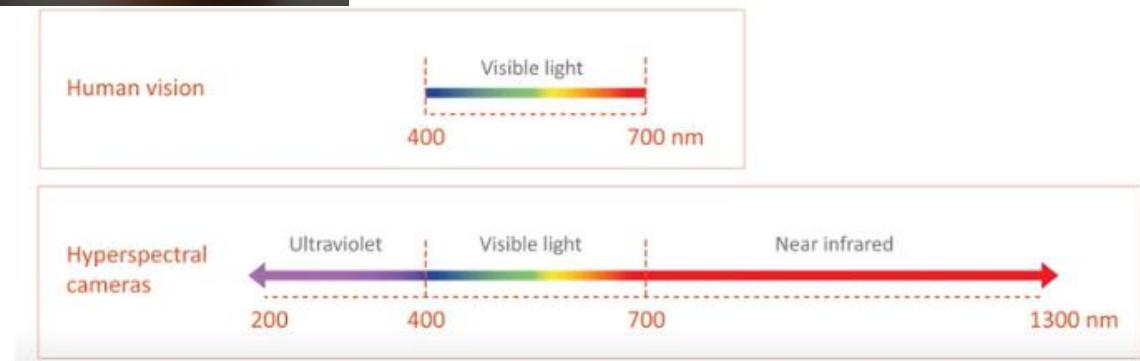
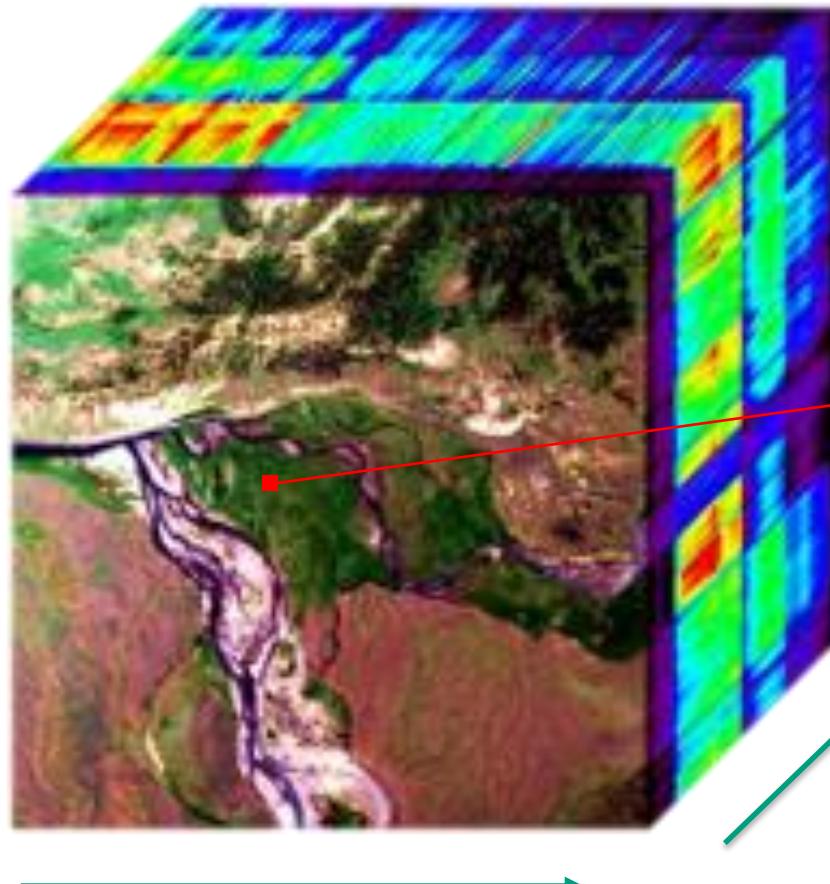
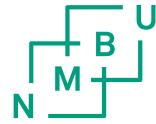
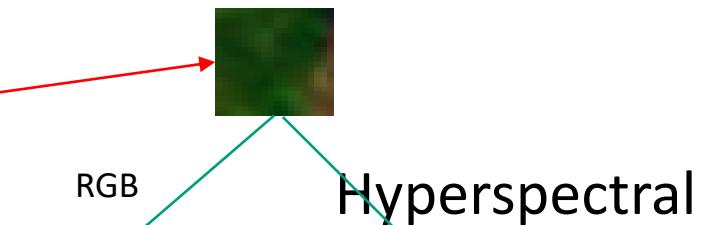


Image Analysis – 3D hyperspectral data cube

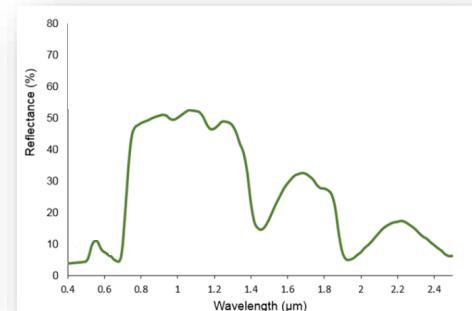


X and Y – spatial

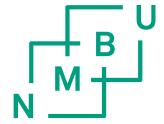
λ – spectral



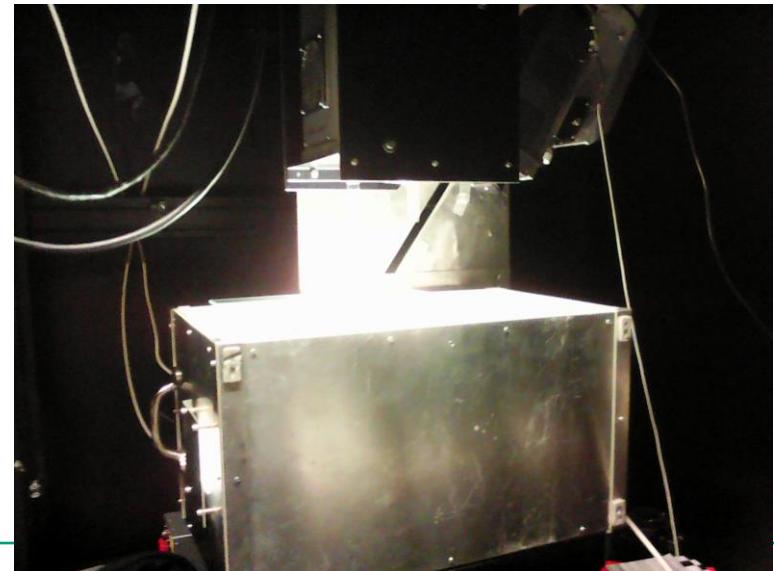
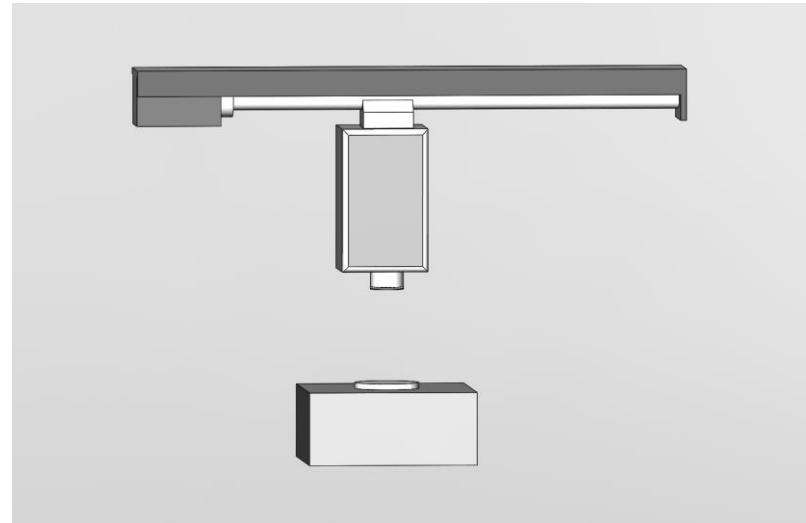
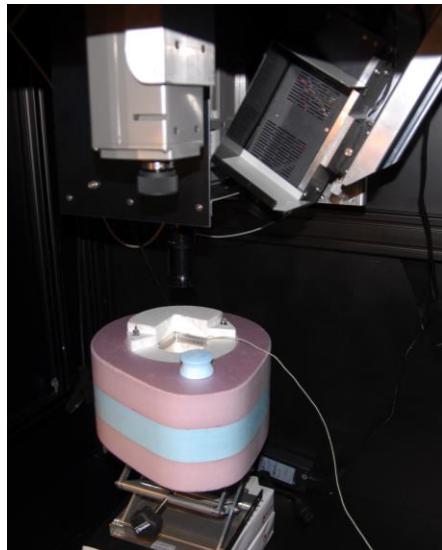
R:0
G:130
B:65



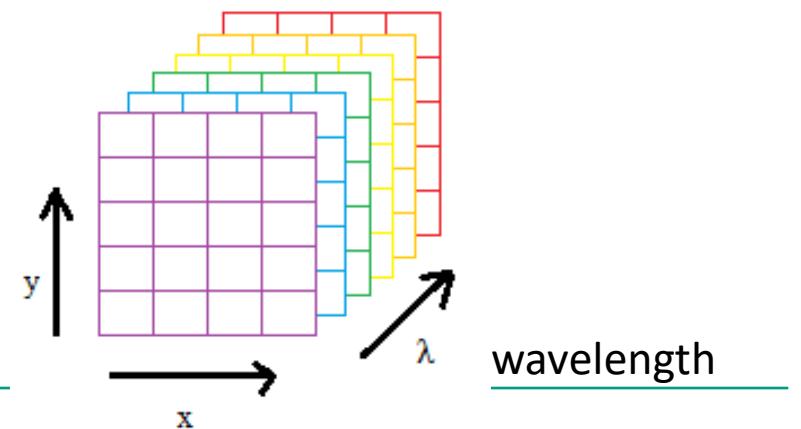
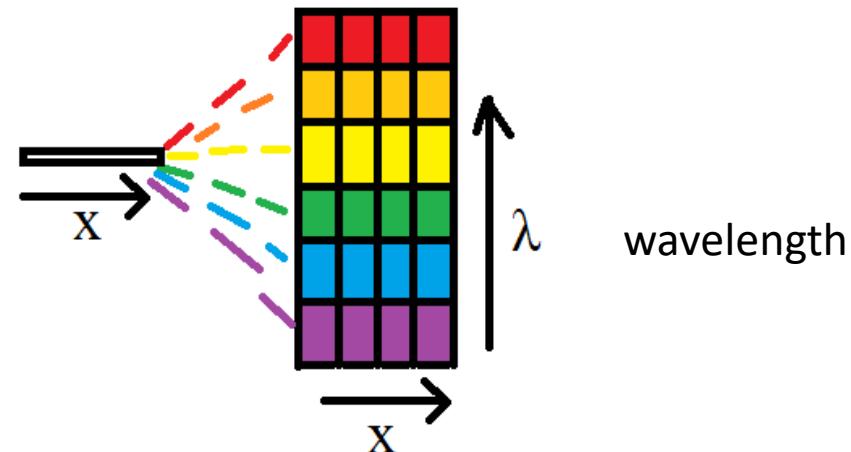
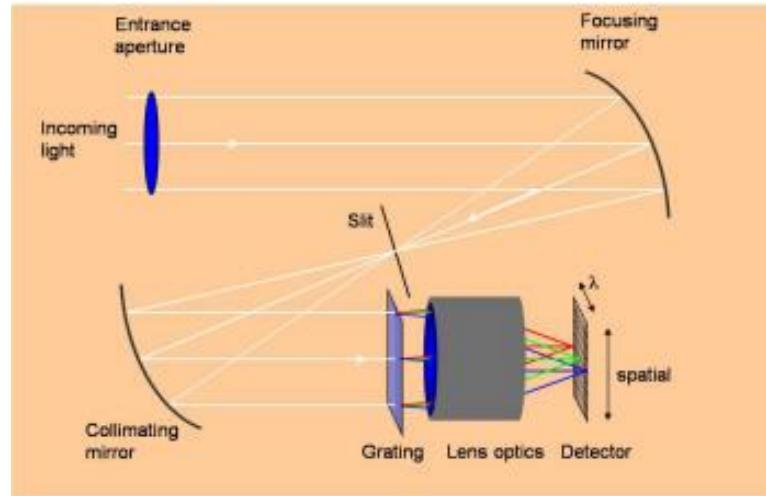
Hyperspectral cameras at Realtek



VIS/NIR 400-1000 nm (Specim)
SWIR 1000-2500 nm (Specim)
SWIR 1000-2500 nm (NEO)



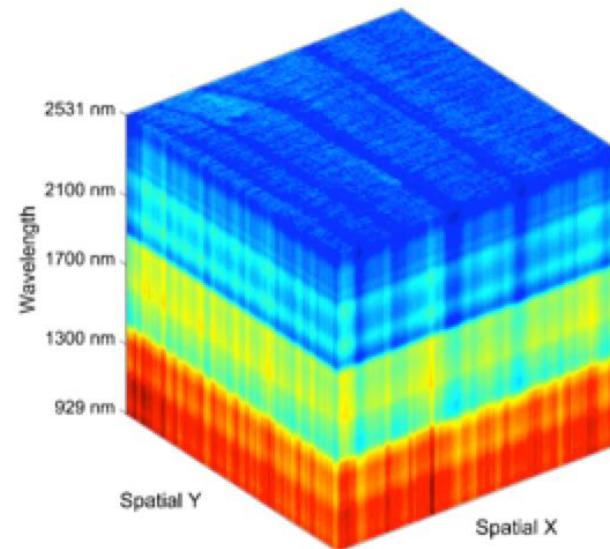
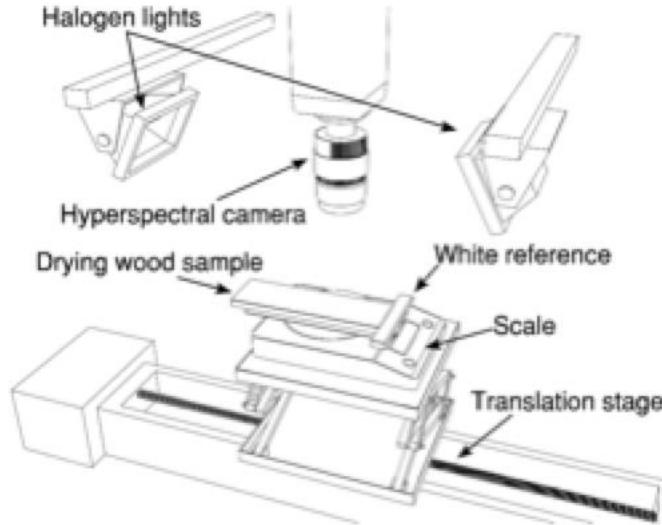
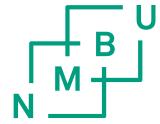
Hyperspectral imaging principle



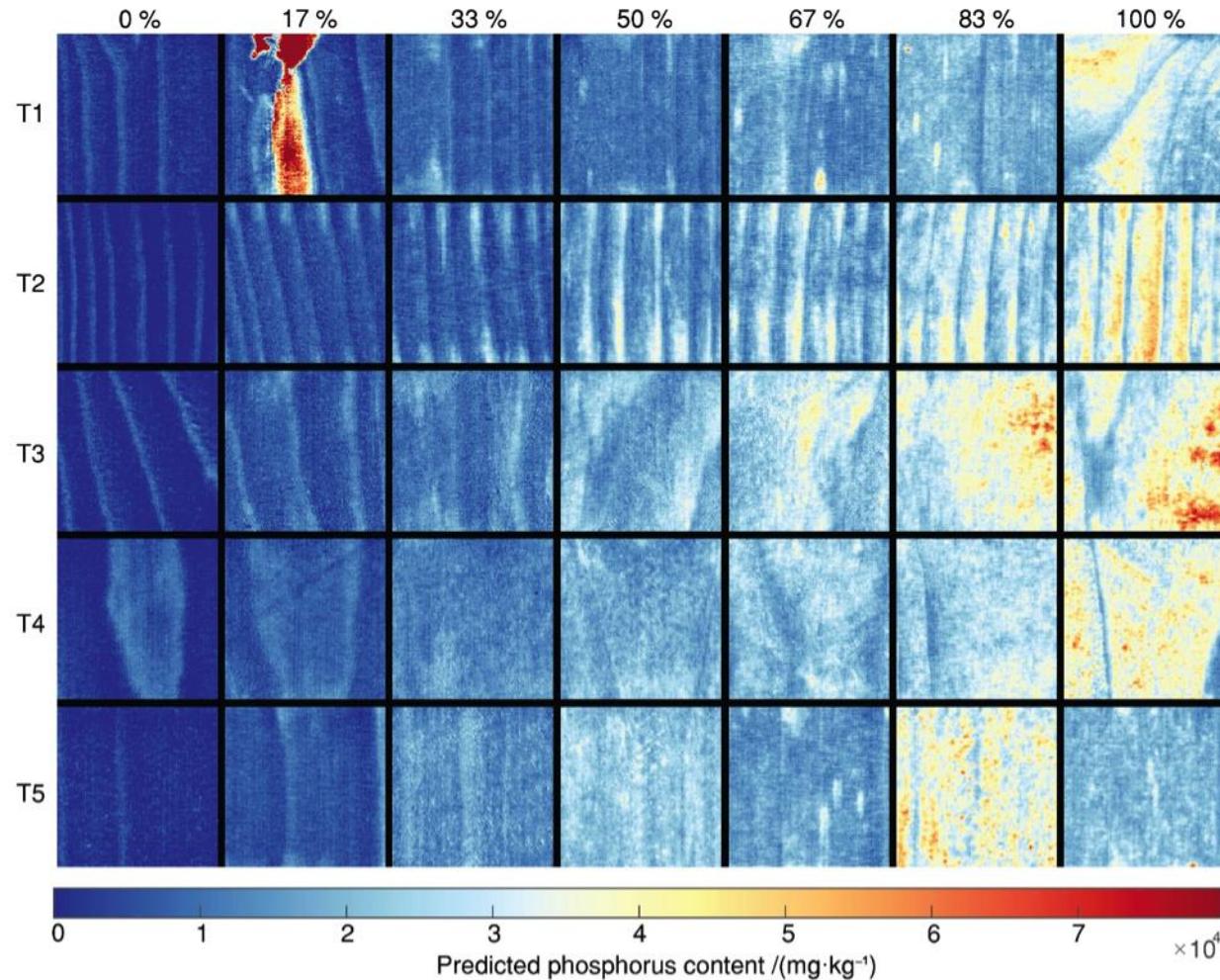
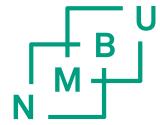
Two types of cameras:

- Line scan camera (bush broom)
- Filter wheel camera

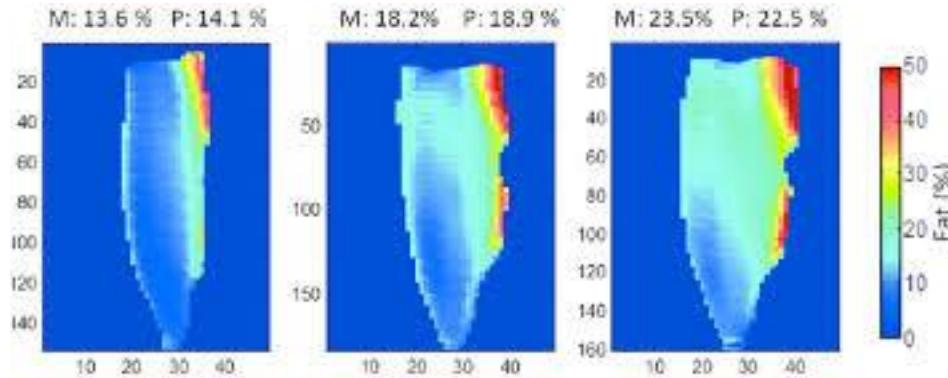
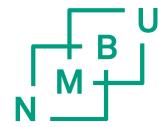
Laboratory measurements



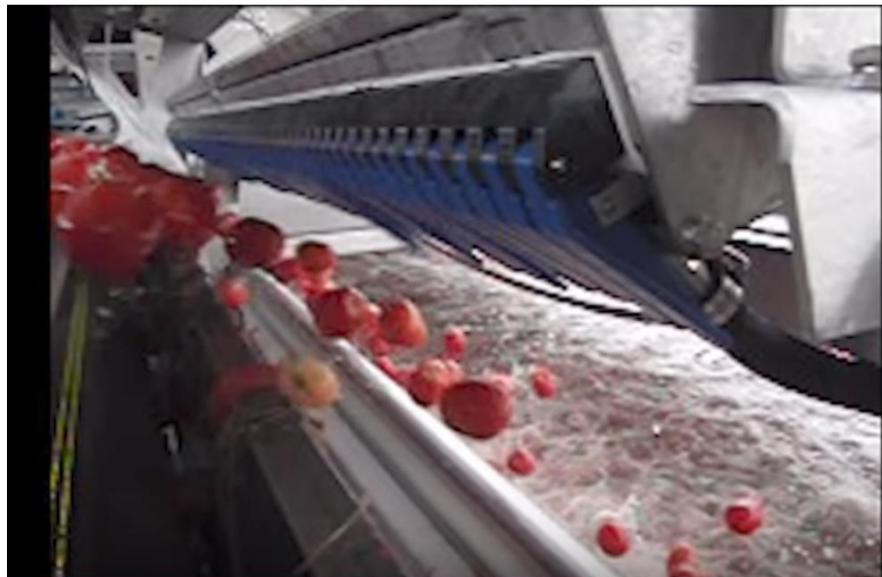
Predicted phosphorus content in wooden surfaces



Applications of hyperspectral imaging



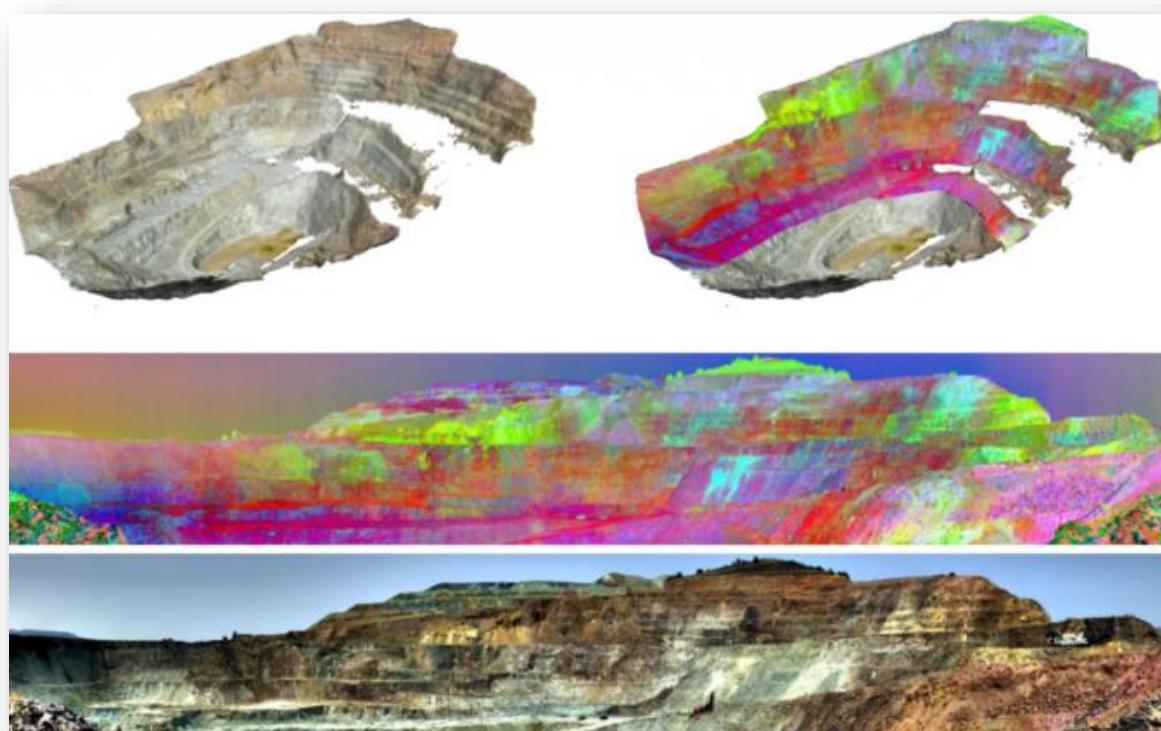
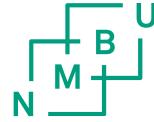
Fat content in salmon, Wold et al. Nofima



Fruit sorting

<https://www.youtube.com/watch?v=Lz88nsWL4k>

Cool applications – Mine industry



- Mineral exploration
- 3D modelling
- Flying drones ☺

Skouriotissa, Cyprus

3D City Modeling



- Urban planning
- Navigation systems
 - 3D models
- Urbanization level

Saving lives



- Searching for aircrafts and saving lives after fatal crashes

Forecasting weather



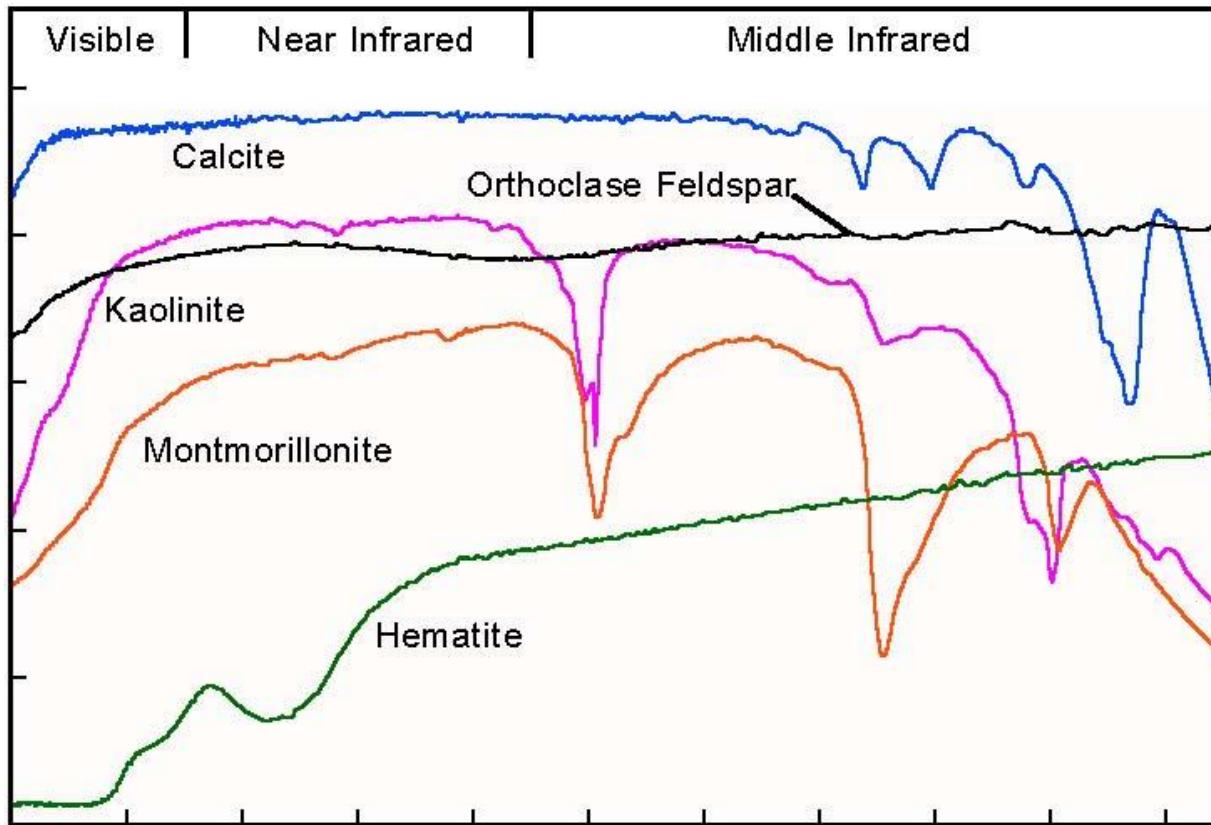
- Forecasting weather to warn about natural disasters

Volcanoes

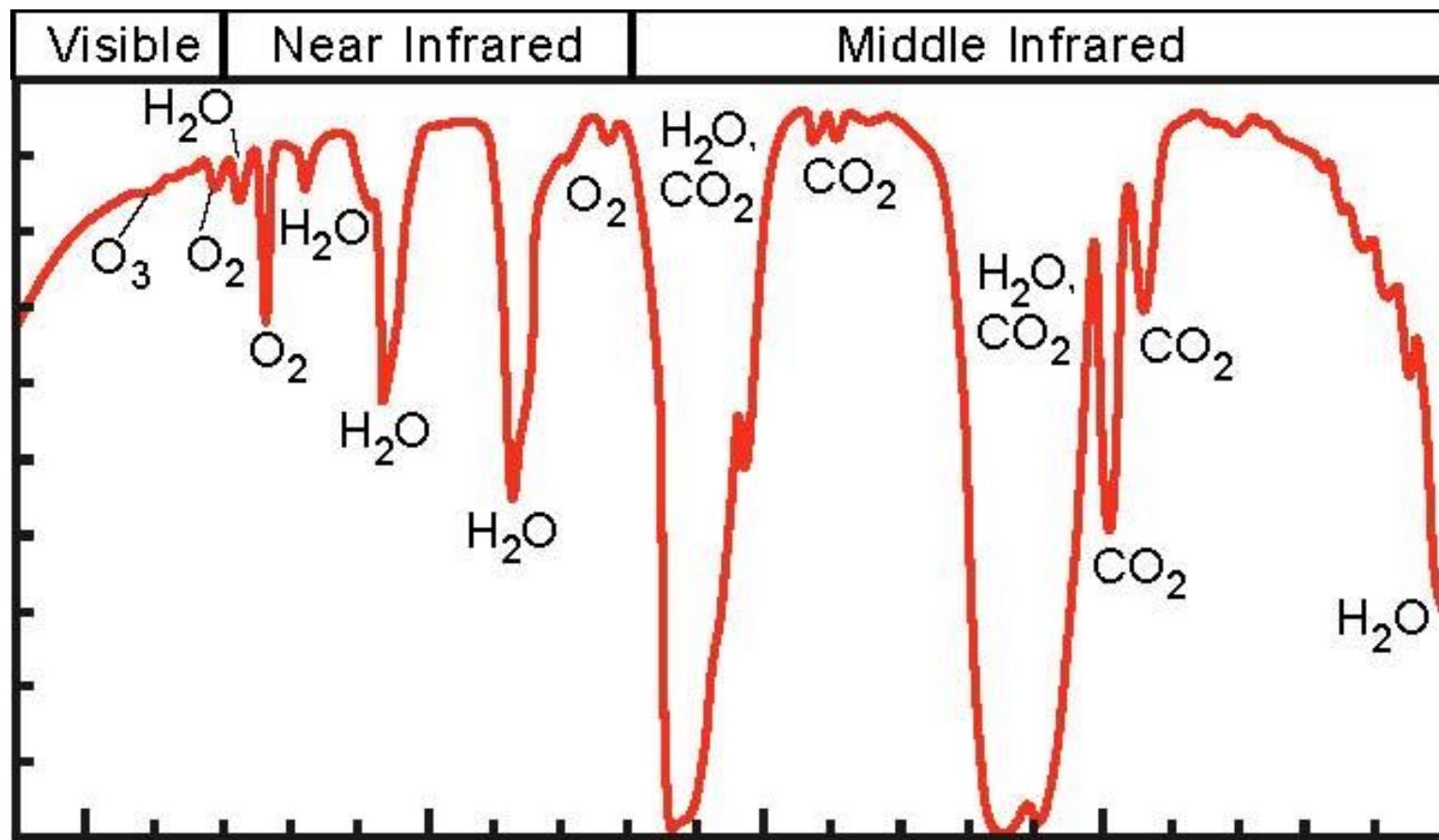


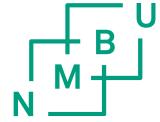
- Monitoring active volcanoes using thermal remote sensing

mineralogy



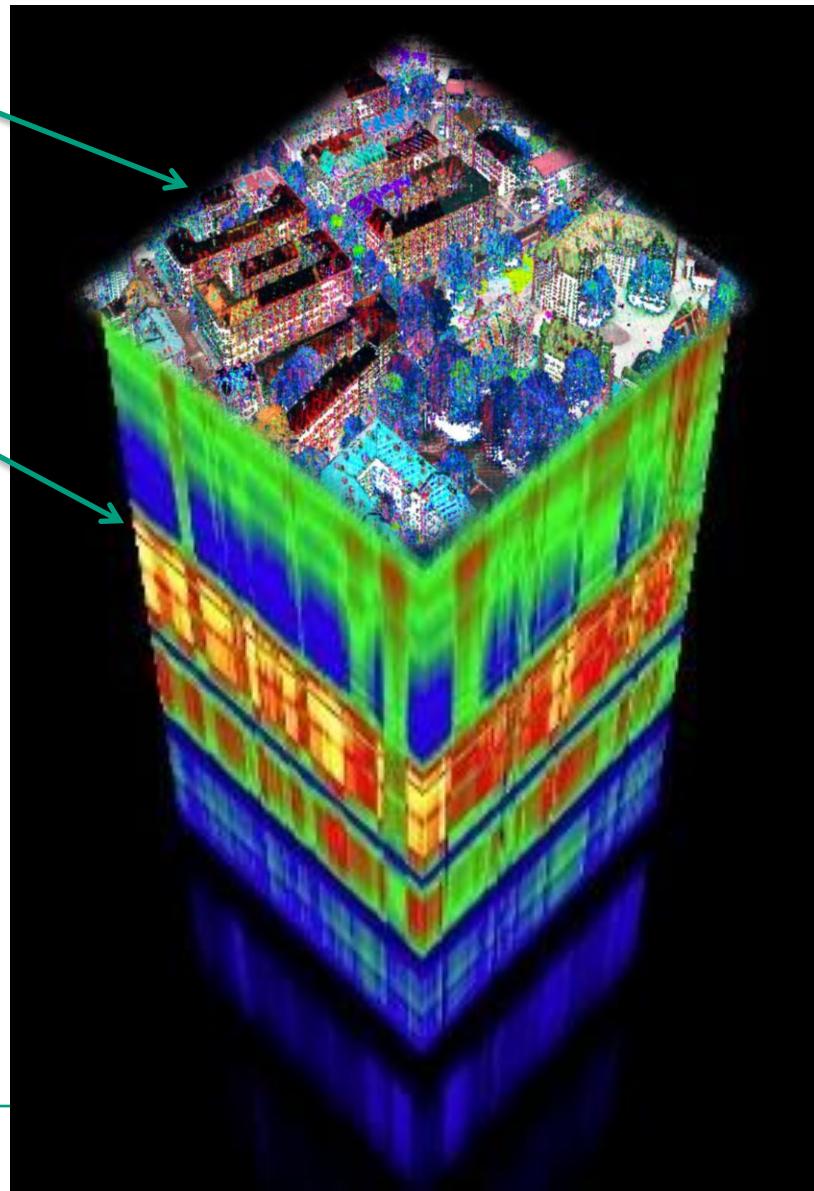
Atmospheric absorption bands





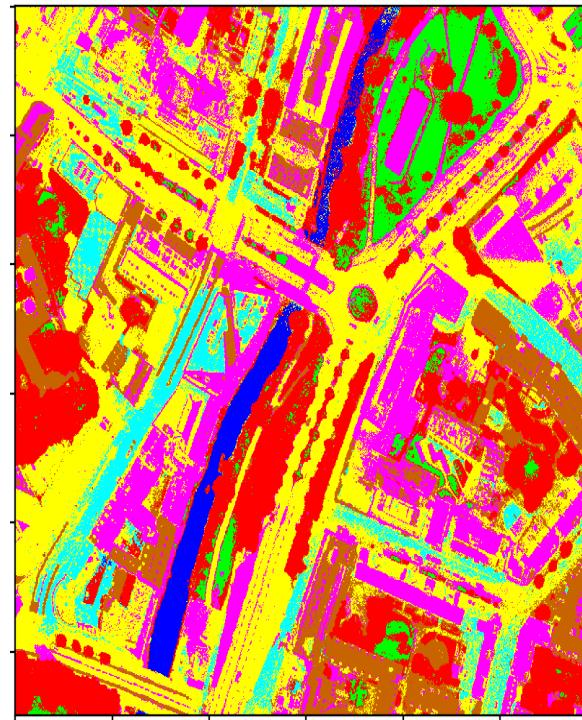
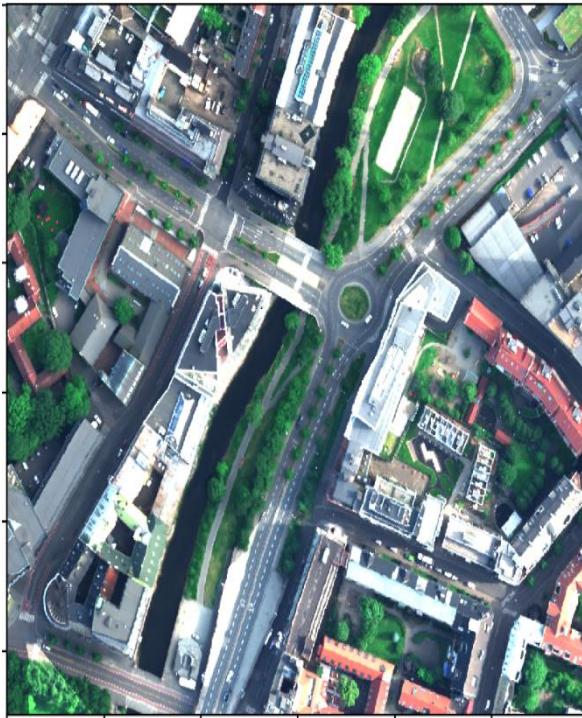
Laser data (LiDAR) yields a 3D model of the surface in the form of a point cloud

Hyperspectral data gives a detailed spectrum of channels in the visible and the near infrared and shortwave infrared, bringing out «invisible» differences in the surface material

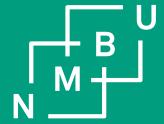


Collaboration with Terratec and
the municipality of Oslo

U
B
M
N

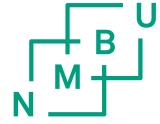


- Asphalt
- Grass
- Trees
- Dark rooftops + sand
- Red rooftops
- Shadowed areas (asphalt)



INF250

Hyperspectral imaging 2



Learning goals

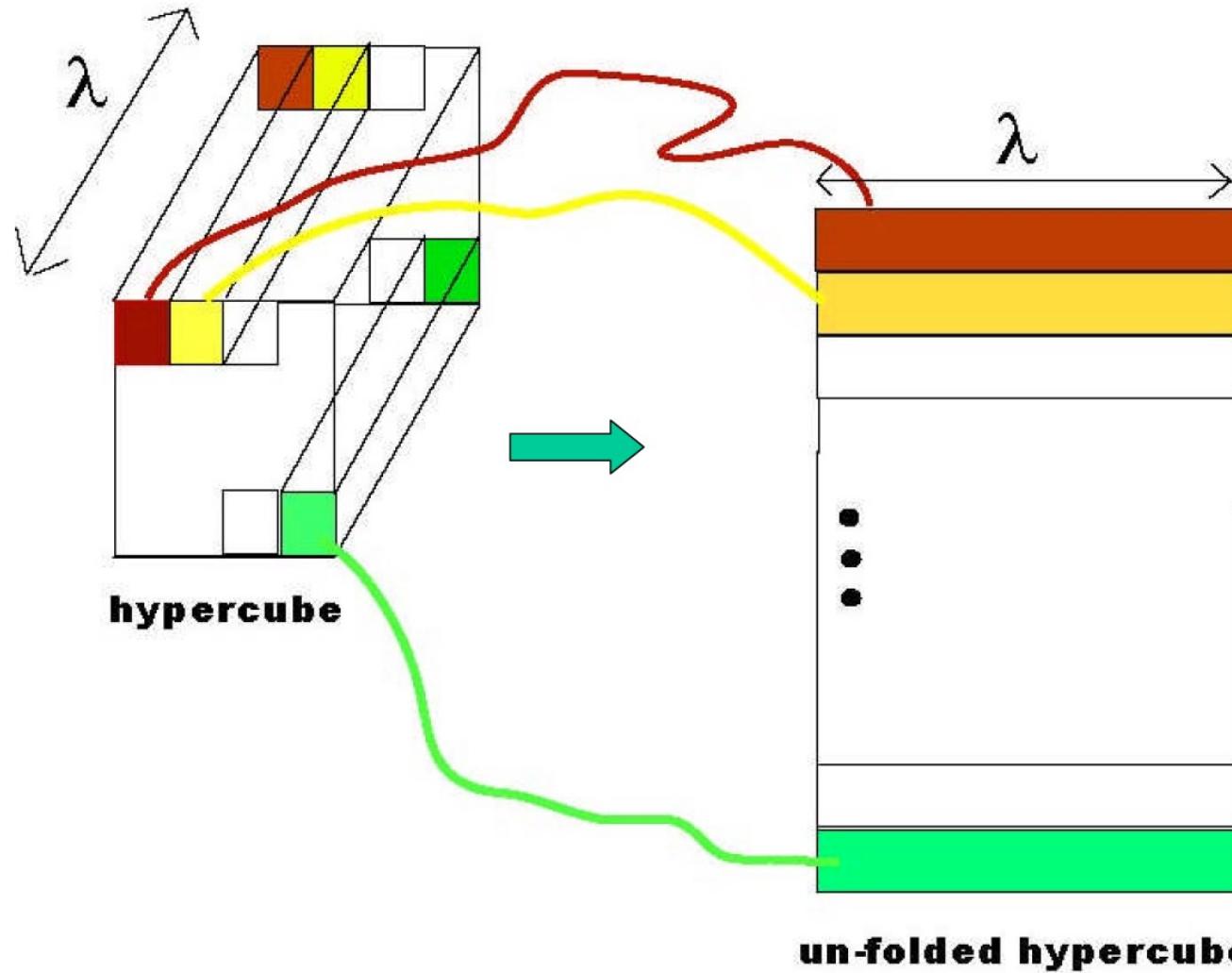
- What is multispectral and hyperspectral imaging ?
- What kind of cameras are used for hyperspectral imaging
- Give som examples of research one can do with hyperspectral imaging
- Describe how PCA works on a hyperspectral image
- Describe how k-means clustering works
- Describe supervised classification



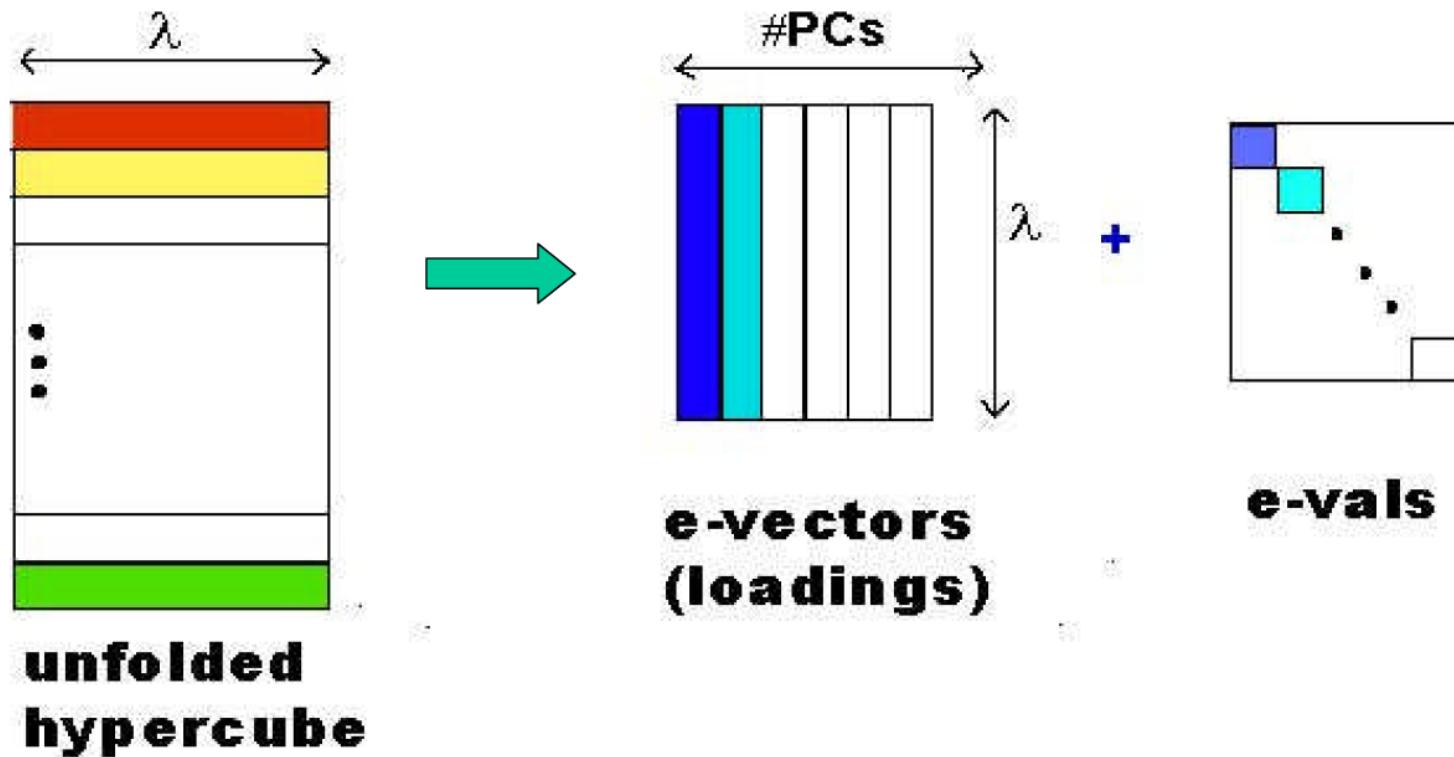
Multivariate image analysis on spectral images^N

- PCA (principal component analysis)
 - K-means clustering
 - Supervised classification
-

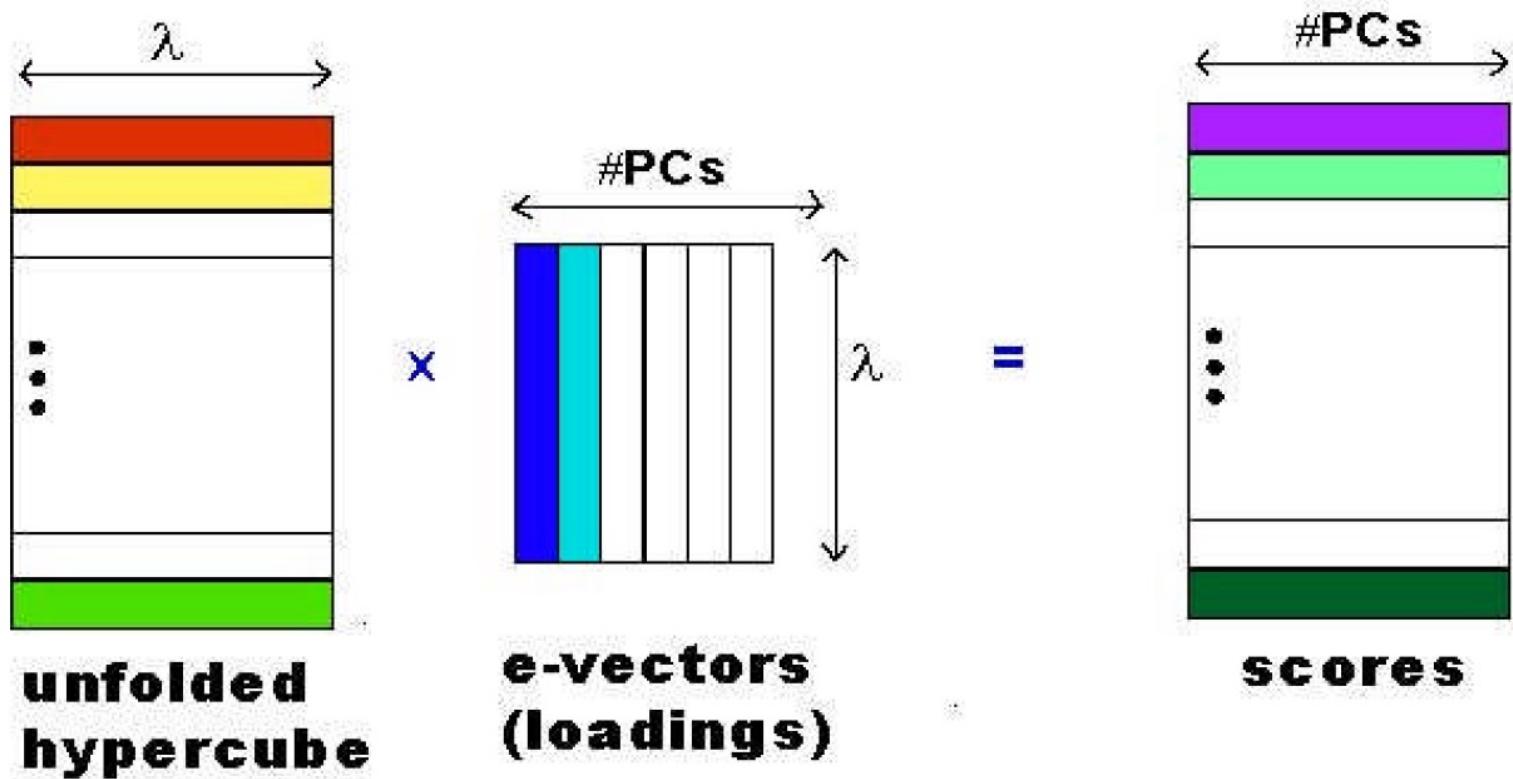
Unfolding the Hypercube



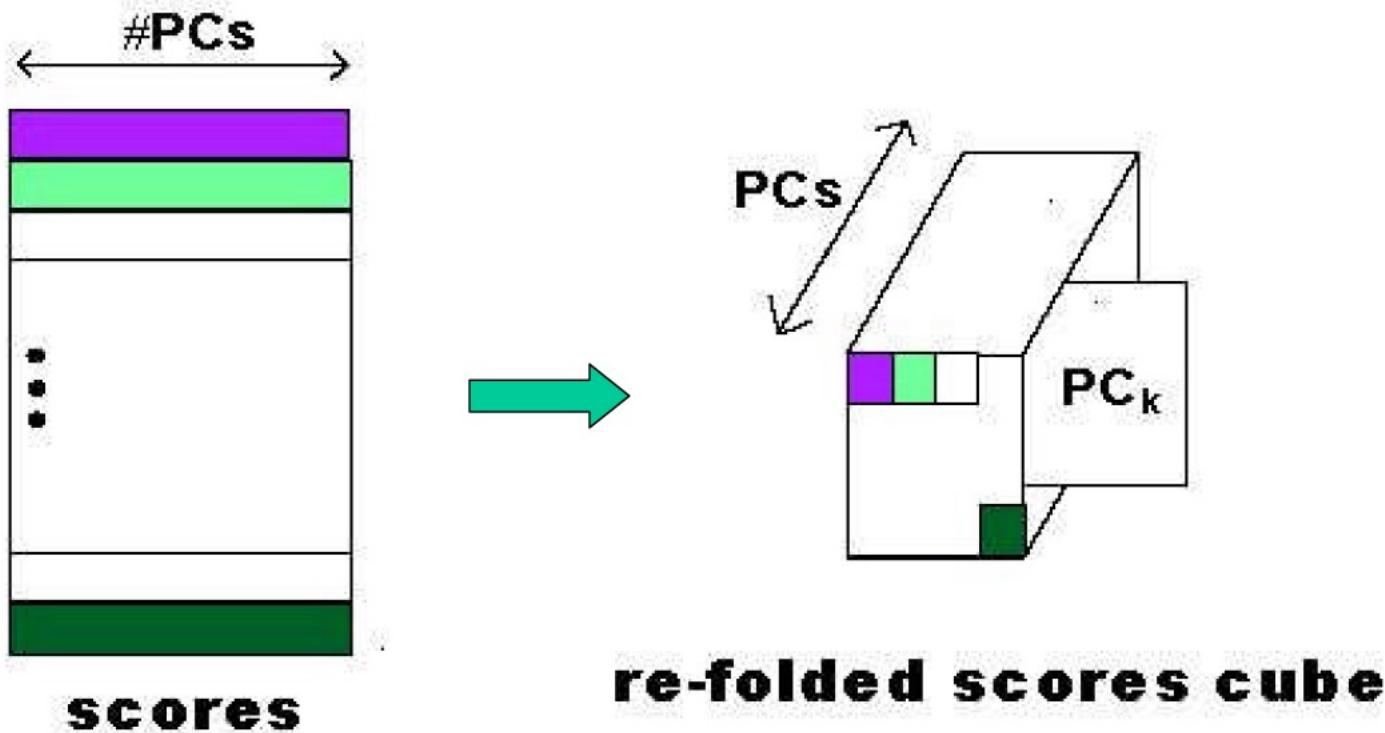
PCA on unfolded hypercube

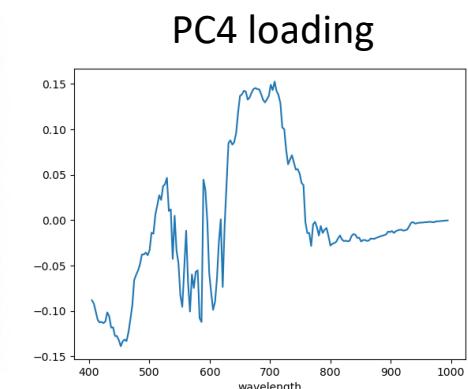
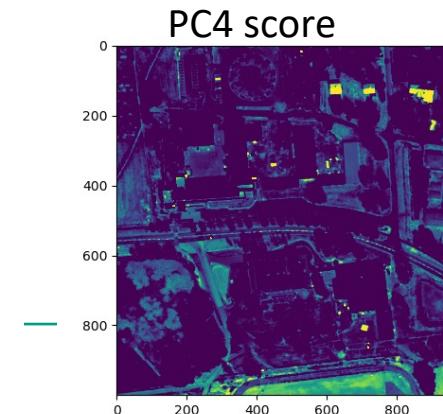
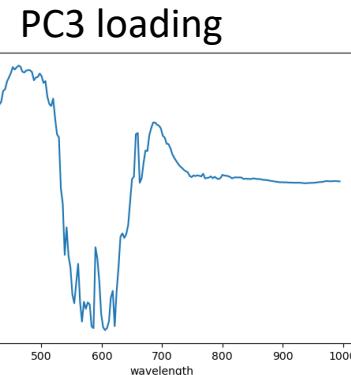
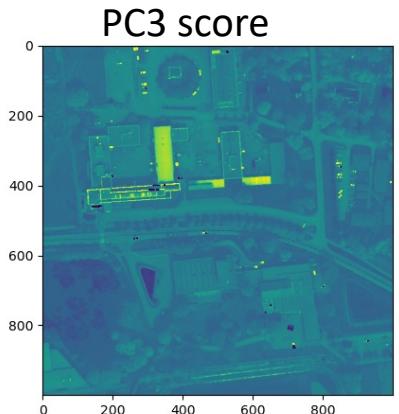
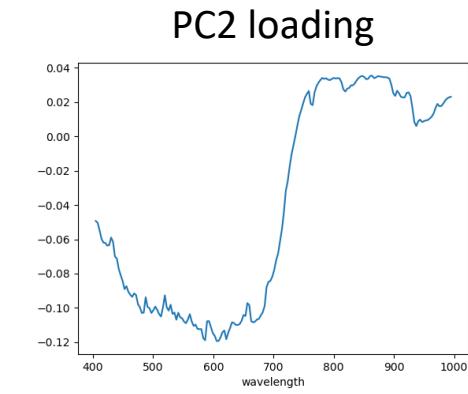
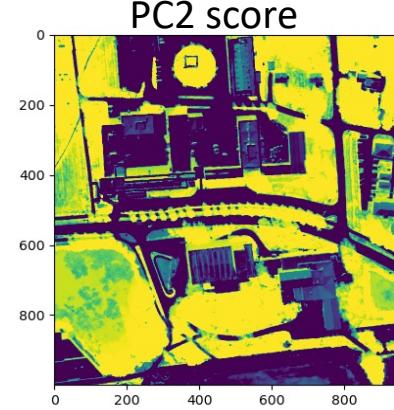
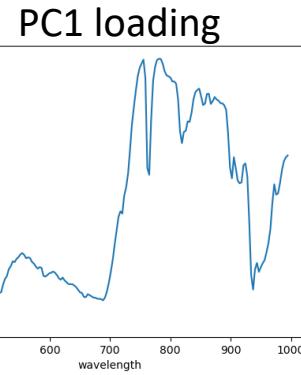
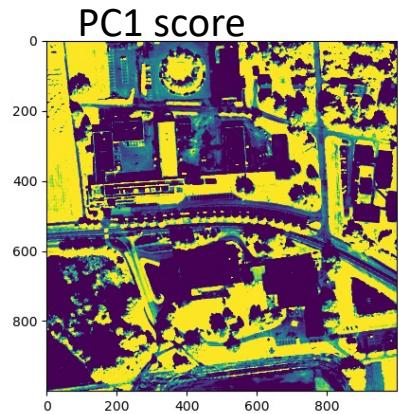
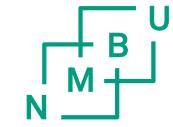
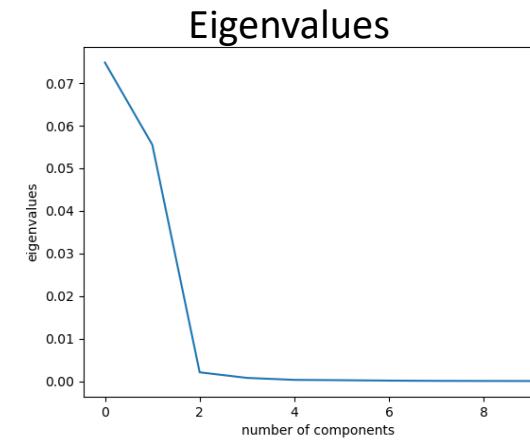
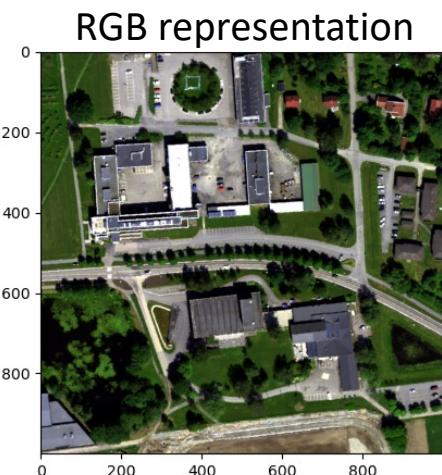


Calculating Scores



Re-folding Scores

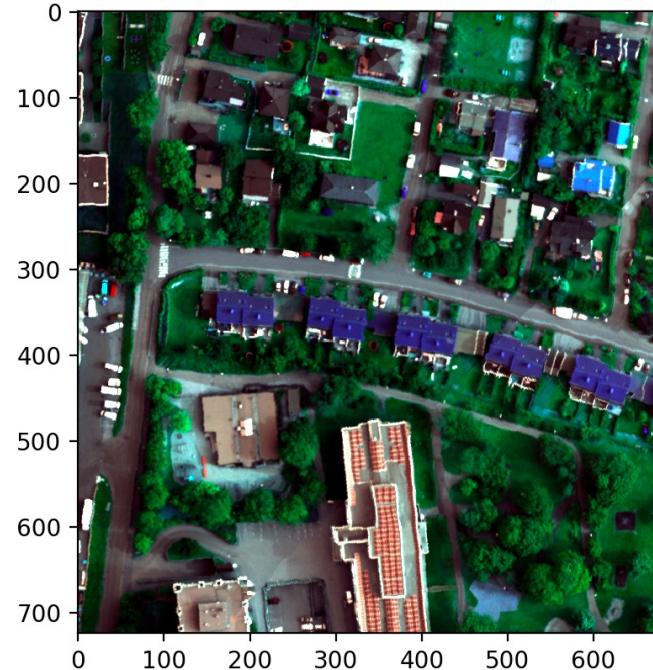




```
from spectral import *
import numpy as np
import matplotlib.pyplot as plt
import skimage

hyperim = np.load("Oslo_hsi.npy")

imshow(hyperim, (10, 48, 70), stretch=((0.02,0.98),(0.02,0.98),(0.02,0.98)))
```



```
z = np.array(hyperim[521,520,:].reshape(-1,1))
z2 = np.array(hyperim[635,214,:].reshape(-1,1))
z3 = np.array(hyperim[441,36,:].reshape(-1,1))
plt.figure()
plt.plot(ww,z)
plt.plot(ww,z2)
plt.plot(ww,z3)
plt.show()

#plot as fct of pixel values
plt.figure()
plt.plot(z)
plt.show()

#compute mean of all spectra
m1 = hyperim.mean(axis=0)
m2 = m1.mean(axis=0).reshape(-1,1)
plt.figure()
plt.plot(ww,m2)
plt.show()

#ndvi

ndvi_ima = (hyperim[:, :, 108] - hyperim[:, :, 94]) / (hyperim[:, :, 108] + hyperim[:, :, 94])
plt.imshow(ndvi_ima, vmin=0, vmax=0.7)

# ndvi from spectral python
vi = ndvi(hyperim, 94, 108)
plt.figure()
plt.imshow(vi, vmin=-0.3, vmax=0.6)
```

```
# pca

pc = principal_components(hyperim)

pc_0999 = pc.reduce(fraction=0.999)

loading = pc_0999.eigenvectors

#loading of pc1
plt.figure()
plt.plot(loading[:,0])

#score of pc1
img_pc = pc_0999.transform(hyperim)
plt.figure()
plt.imshow(img_pc[:, :, 0], vmin=-0.1, vmax=0.15)
```

```
# kmeans

(m, c) = kmeans(hyperim, 7, 20)
plt.imshow(m, 'spectral')

plt.figure()
for i in range(c.shape[0]):
    plt.plot(c[i])
```



```
shape = hyperim.shape
groundtruth = np.zeros([shape[0],shape[1]])

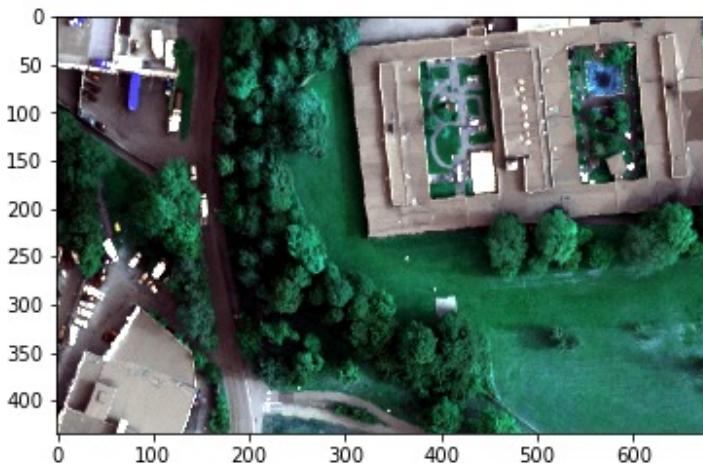
groundtruth[128:168, 336:366] = 1.0    #grass
groundtruth[623:650, 200:250] = 2.0 # asphalt
groundtruth[460:480, 150:170] = 3.0 # roof1

groundtruth[350:370, 300:320] = 3.0 # roof1

plt.figure()
plt.imshow(groundtruth)
```

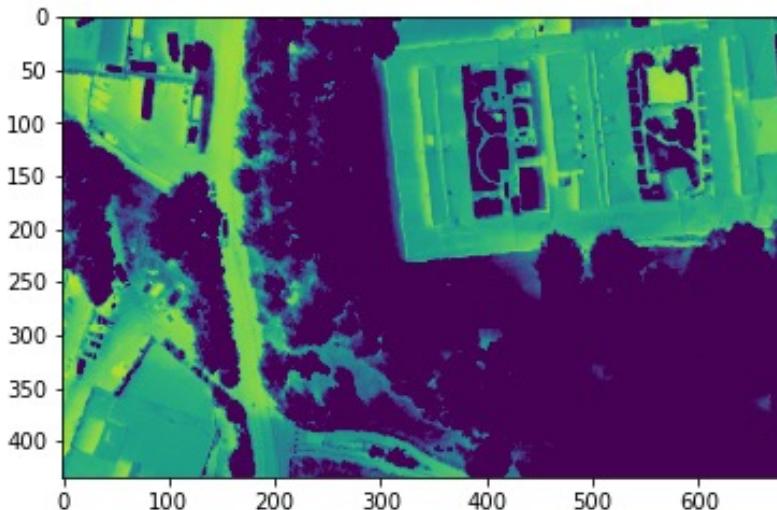
Gaussian Maximum Likelihood classification

```
classes = create_training_classes(hyperim, groundtruth)
gmlc = GaussianClassifier(classes)
clmap = gmlc.classify_image(hyperim)
imshow(classes=clmap)
```

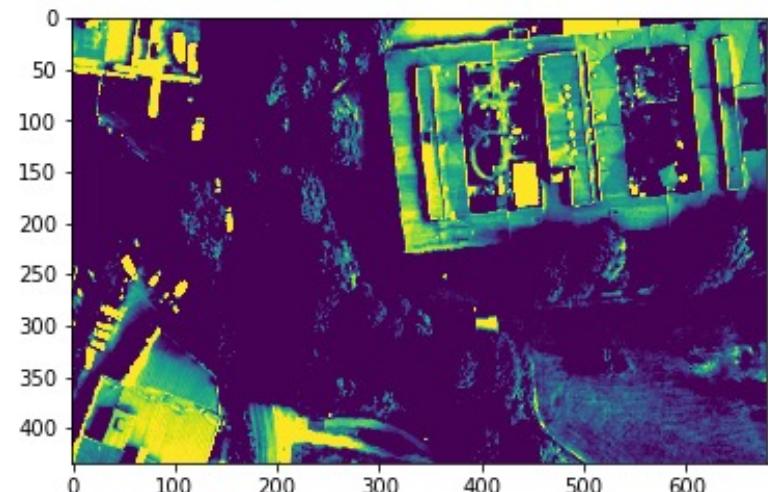


```
# pca  
pc = principal_components(hyperim)  
pc_0999 = pc.reduce(fraction=0.999)  
len(pc_0999.eigenvalues)  
img_pc = pc_0999.transform(img)  
plt.imshow(img_pc[:, :, 0], vmin=0.0, vmax=0.3)  
plt.imshow(img_pc[:, :, 1], vmin=0.0, vmax=0.1)
```

Hyperspectral image composite
of band 20,40 and 60

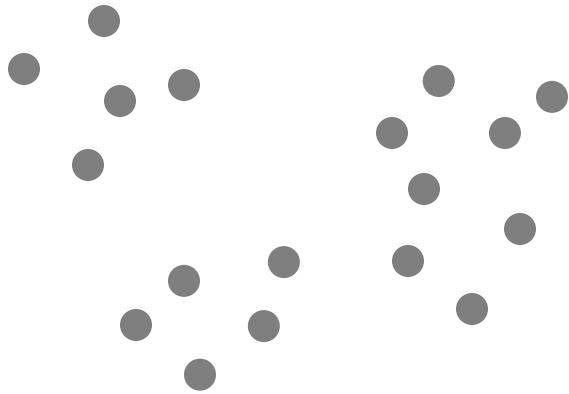
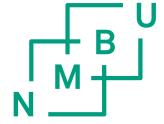


PC 1

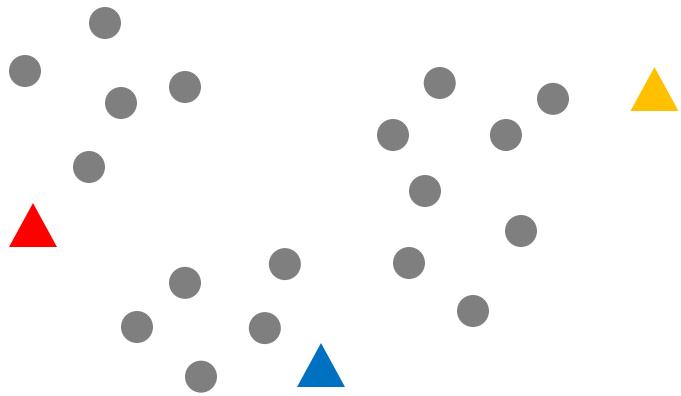
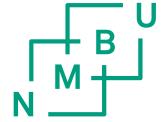


PC 2

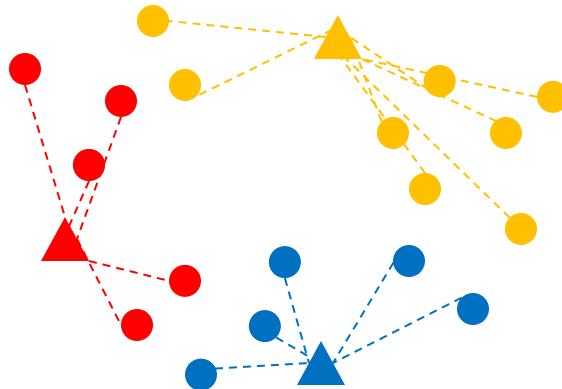
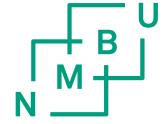
Clustering: K-means



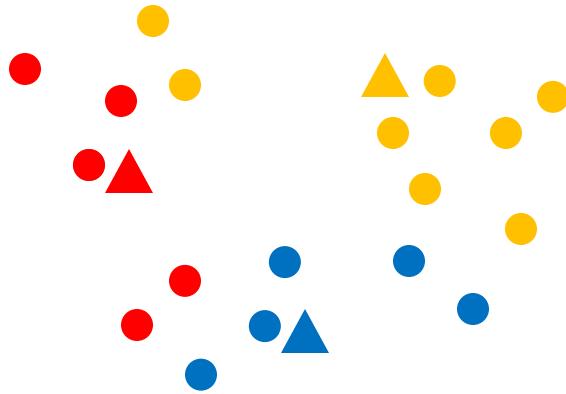
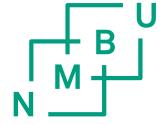
Clustering: K-means



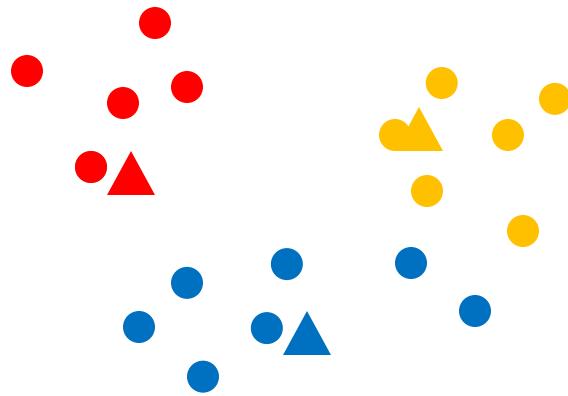
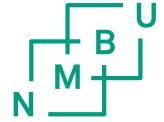
Clustering: K-means



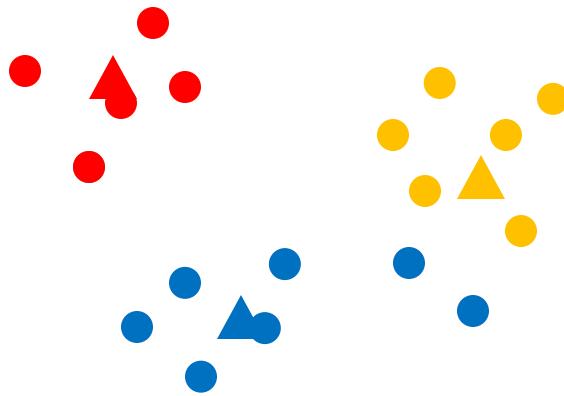
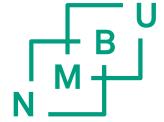
Clustering: K-means



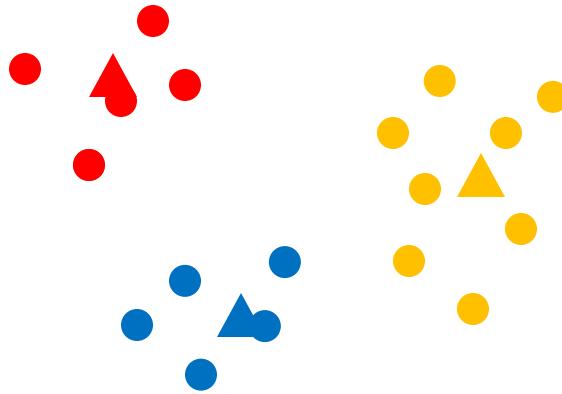
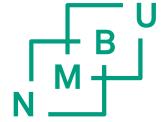
Clustering: K-means



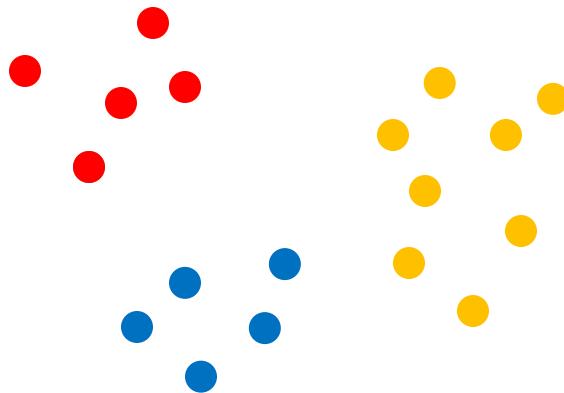
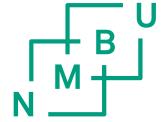
Clustering: K-means



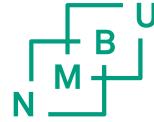
Clustering: K-means



Clustering: K-means



Clustering



- K-means clustering

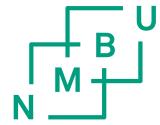
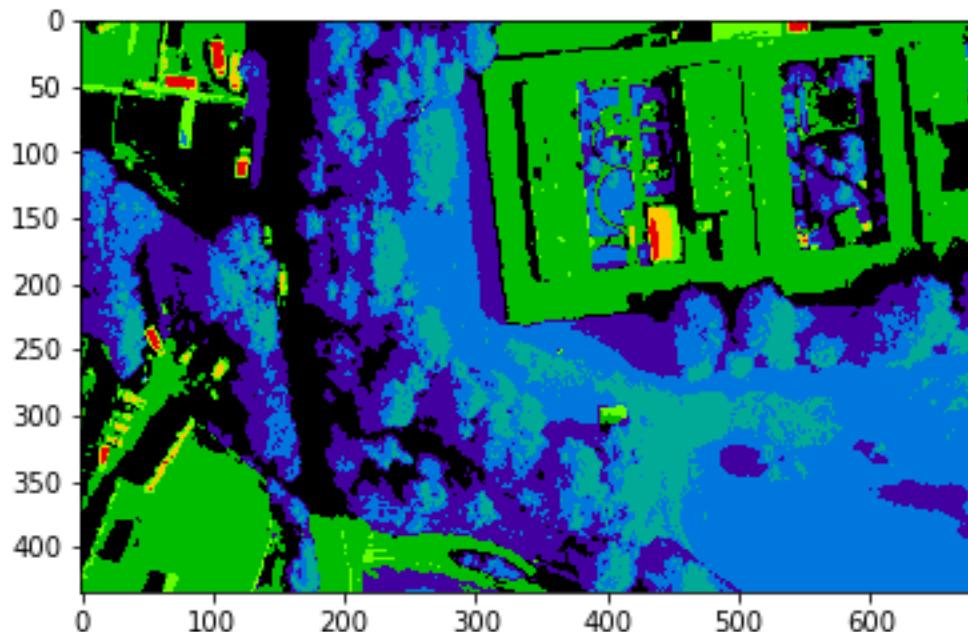
- Separates samples into K groups of equal variance

- 1. Initialization:** Choose K initial cluster centroids randomly from the data points

- 2. Assignment:** Assign each data point to the nearest centroid, forming K clusters. The distance between data points and centroids is typically measured using Euclidean distance

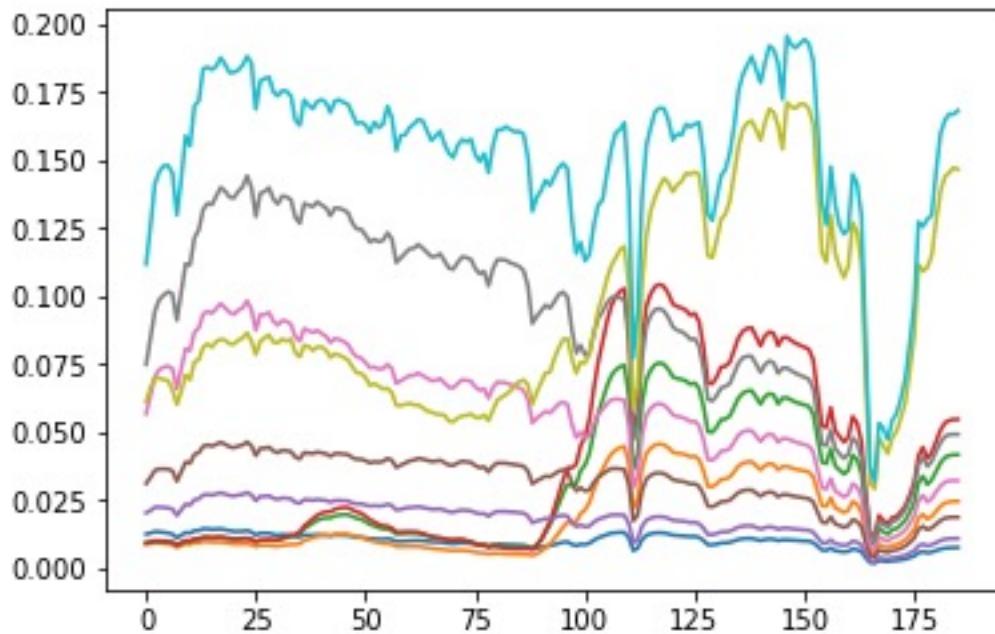
- 3. Update centroids:** Calculate the mean of the data points in each cluster and update the centroid to be the mean. This moves the centroids to the center of their respective clusters

- 4. Repeat:** Repeat steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or after a fixed number of iterations



```
# kmeans
(m, c) = kmeans(hyperim, 10, 30)
plt.figure()
plt.imshow(m, 'spectral')
plt.show()

for i in range(c.shape[0]):
    plt.plot(c[i])
```



Classification

Assigning predefined labels to objects based on their features



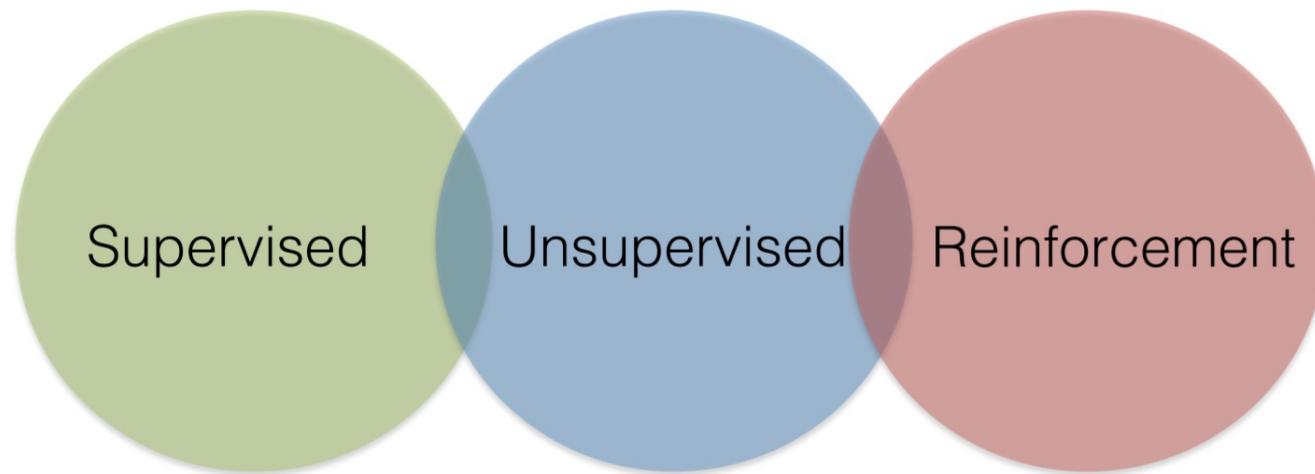
Clustering



Grouping a set of objects in such a way that objects in the same group are more similar in some sense than objects in other groups



Classification in images



Learning from
“labelled” data

Discover structure
in “unlabelled” data

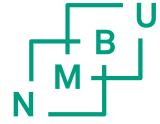
Learning by “doing”
with delayed reward

Non supervised vs supervised classification



Unsupervised classification is where the outcomes (groupings of pixels with common characteristics) are based on the software analysis of an image without the user providing sample classes. The computer uses techniques to determine which pixels are related and groups them into classes. The user can specify which algorithm the software will use and the desired number of output classes but otherwise does not aid in the classification process. However, the user must have knowledge of the area being classified when the groupings of pixels with common characteristics produced by the computer have to be related to actual features on the ground (such as wetlands, developed areas, coniferous forests, etc.).

Supervised classification is based on the idea that a user can select sample pixels in an image that are representative of specific classes and then direct the image processing software to use these training sites as references for the classification of all other pixels in the image. Training sites (also known as testing sets or input classes) are selected based on the knowledge of the user. The user also sets the bounds for how similar other pixels must be to group them together. These bounds are often set based on the spectral characteristics of the training area, plus or minus a certain increment (often based on "brightness" or strength of reflection in specific spectral bands). The user also designates the number of classes that the image is classified into. Many analysts use a combination of supervised and unsupervised classification processes to develop final output analysis and classified maps.



Examples of supervised classification

- Gaussian Maximum Likelihood Classifier (GMLC)
- Spectral Angle Mapper (SAM)
- Discriminant Analysis

Gaussian Maximum Likelihood Classifier (GMLC)

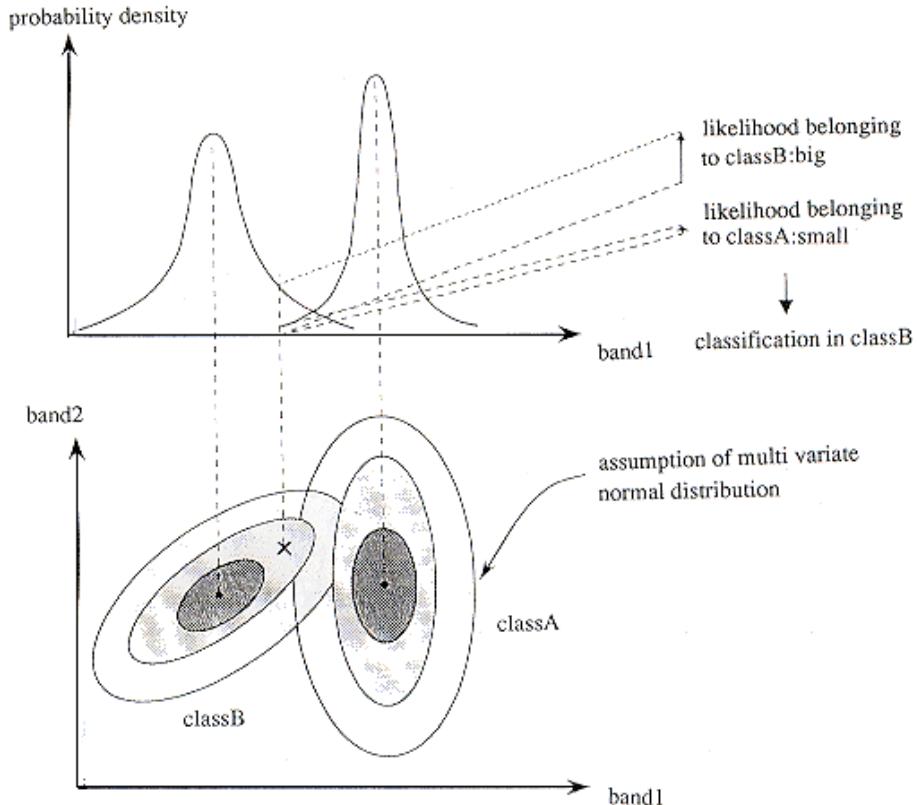
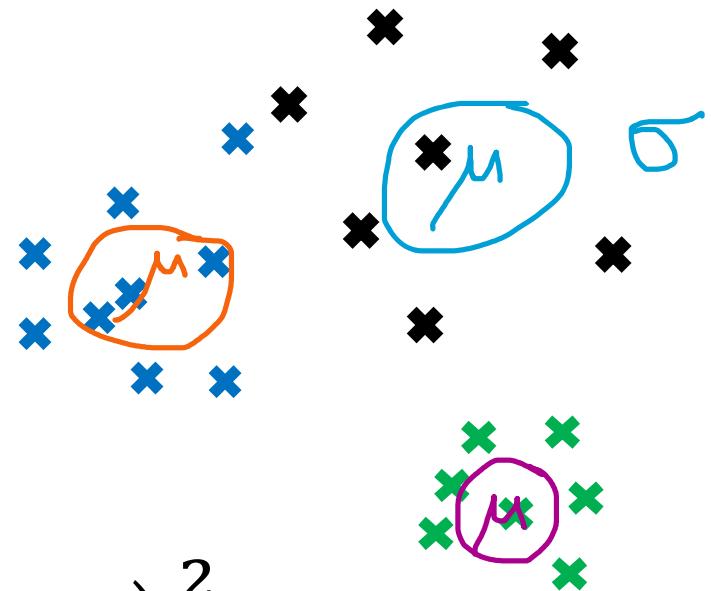


Figure 11.7.1 Concept of Maximum Likelihood Method

Gaussian Maximum Likelihood Classifier (GMLC)

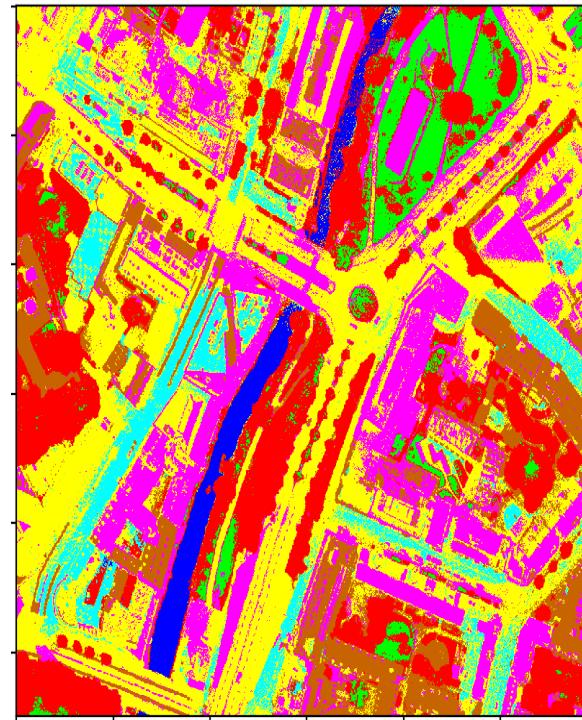
Assume that the probability density follows a gaussian distribution with a mean μ and standard variation σ

Compute the maximum likelihood that a new point belongs to a class.



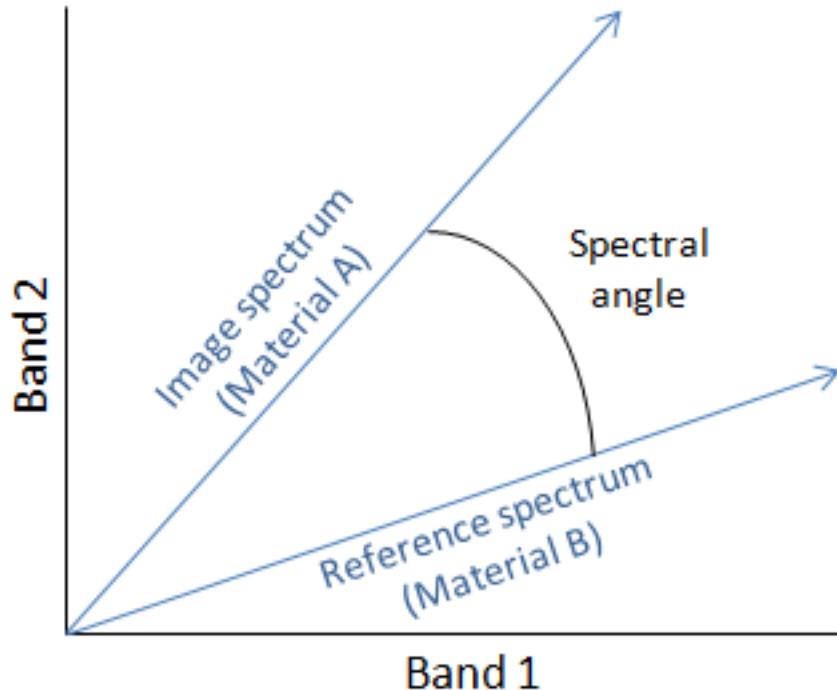
$$p(x|\omega_j) = \frac{1}{\omega_j} \sqrt{2\pi} e^{-\frac{1}{2} \left(\frac{x - \mu_j}{\sigma_j} \right)^2}$$

Maximum Likelihood Classification



- Asphalt
- Grass
- Trees
- Dark rooftops + sand
- Red rooftops
- Shadowed areas (asphalt)

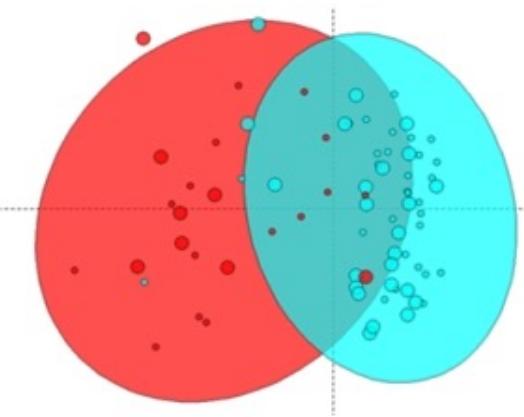
Spectral Angle Mapper



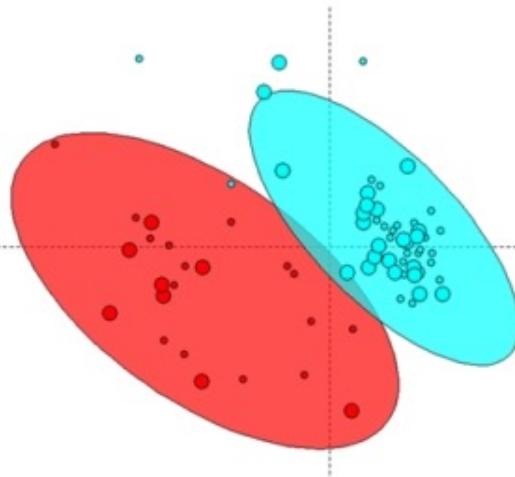
PLS-DA

Rotation of PCA score plot so as to separate classes

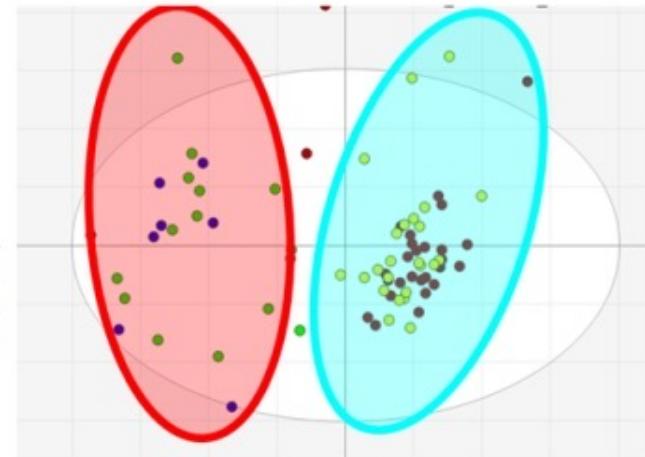
PCA



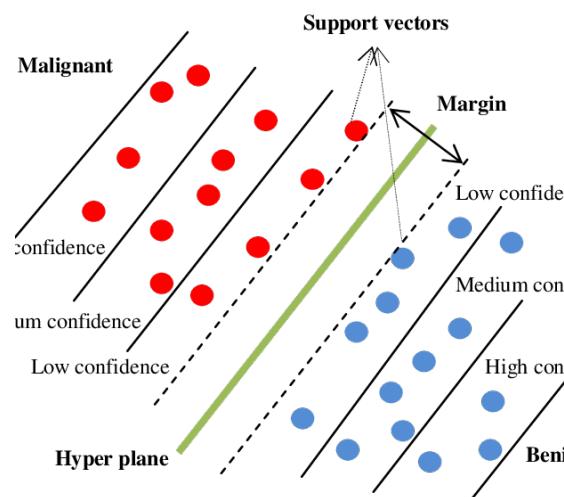
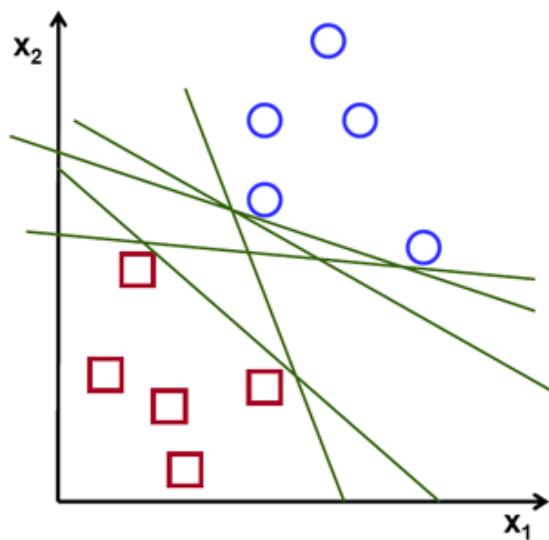
PLS-DA



O-PLS-DA



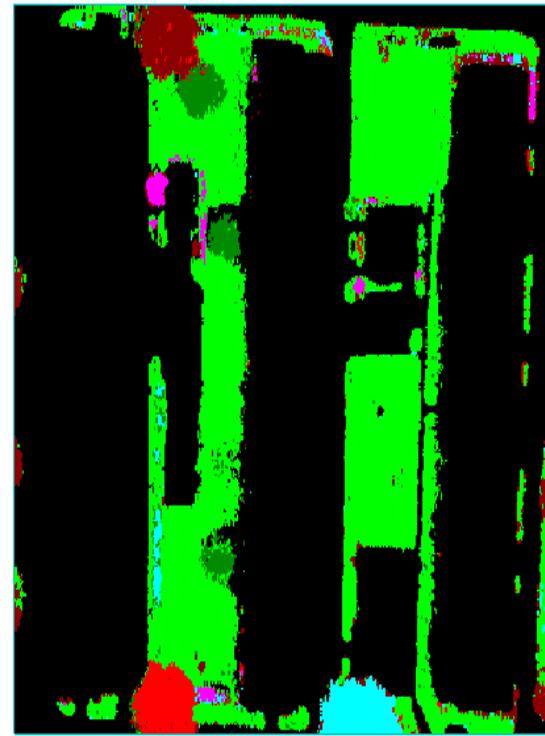
Support Vector Machine



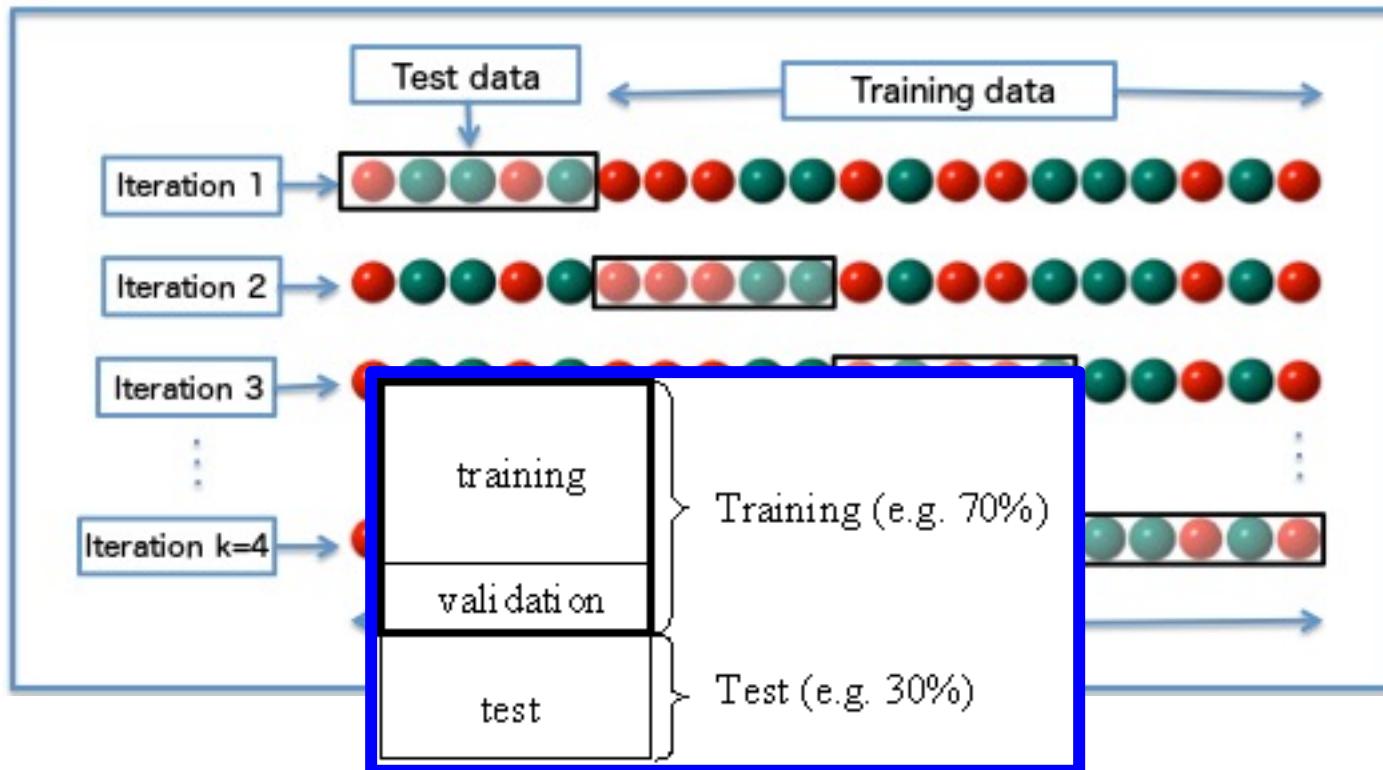
Classification of tree species

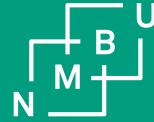
Different tree species have different leaf area, which is related to capacity of absorb polluting particles in the air

Support Vector machine classification



Validation



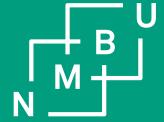


Welcome to INF250

Image Analysis

Ingunn Burud

Ingunn.burud@nmbu.no



About me ...

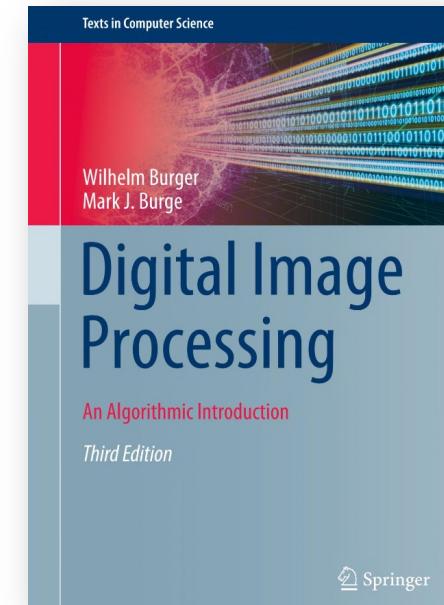
PhD in astrophysics

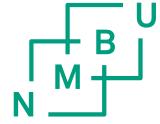
Professor in Physics at Realtek

- vegetation analyses
- material characterization
- Solar cell imagery
- remote sensing

Book

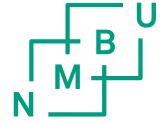
- Burger&Burge: Digital Image Processing
 - An Algorithmic Introduction Using Java 2. Edition
 - <http://imagingbook.com>
- Some extra material
- Plans for the course on Canvas
 - Will be updated





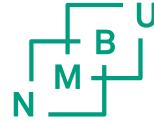
The team

- Agnieszka Kuras, lecturer and course responsible
- Email: Ingunn.burud@nmbu.no
- Sindre Stokke, assistant
- Lavanyan Rathy, assistant
- Lectures
 - Tuesdays 14:15-16 (TF1-102)
- Exercises
 - Thursdays 12:15-18 (TF4-102)



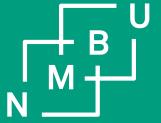
Software and resources

- Python (Anaconda)
- <https://www.anaconda.com/download/>
 - Scikit-image <http://scikit-image.org/>
- ImageJ / Fiji
- <https://fiji.sc/>



The work

- You will use your own PC/Mac
- Theory and techniques will be presented in the lectures
- There will be exercises related to each lecture
 - If needed we will go through some exercises together
- The exercises are a big part of the course, it is very important that you carry out all the exercises for your own understanding of the material
- There will be 3 exercises to hand in, where 2 need to be approved
- The exercises are a part of the course and considered known for the exam



Introduction to Deep Learning for image analysis

Sahameh Shafiee



What you expect to learn after this lecture

- You will be able to understand structure of neural networks and how they are working
- You will learn about different neural networks
- You will get familiar with deep learning and how to apply it for prediction or image classification



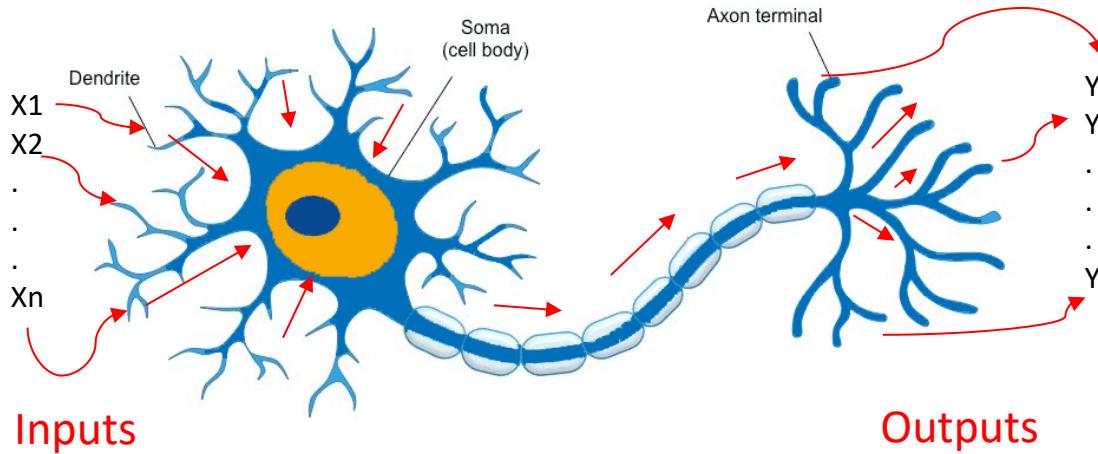
What are artificial neural networks?

- An artificial neural network represents the structure of a human brain modeled on the computer. It consists of neurons and synapses organized into layers.
- ANN can have millions of neurons connected into one system, which makes it extremely successful at analyzing and even memorizing various information.

Biological neuron and artificial neural network

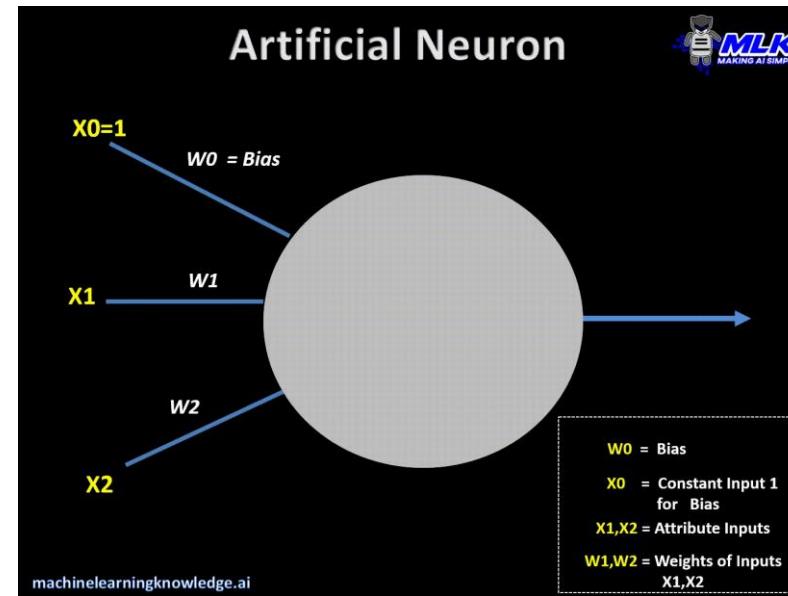


A biological neuron, also known as a nerve cell, is a cell that takes in, processes and transmits information through electrical and chemical signals. It is one of the basic elements of our nervous system.



<https://adatis.co.uk/introduction-to-artificial-neural-networks/>

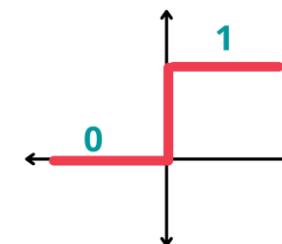
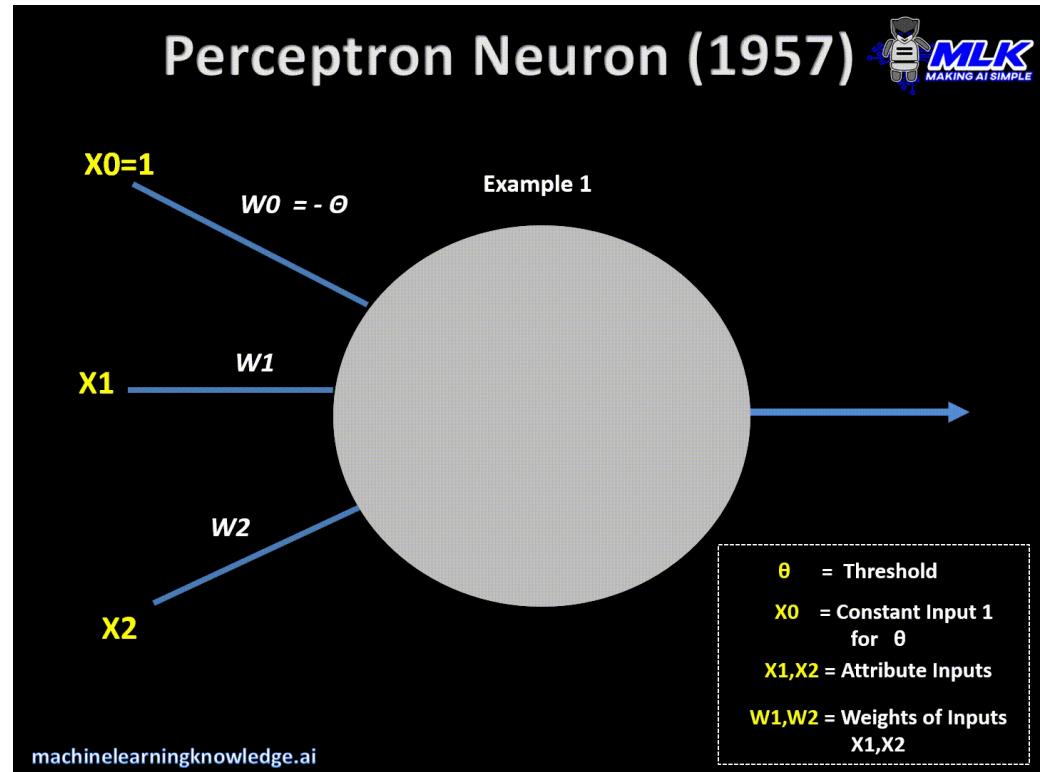
The artificial neuron is modeled based on the biological one and can be represented as a mathematical function or a single processing unit that is part of a network. As with the biological neuron, the artificial one is also connected to other neurons in the network.



<https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/>

Perceptron

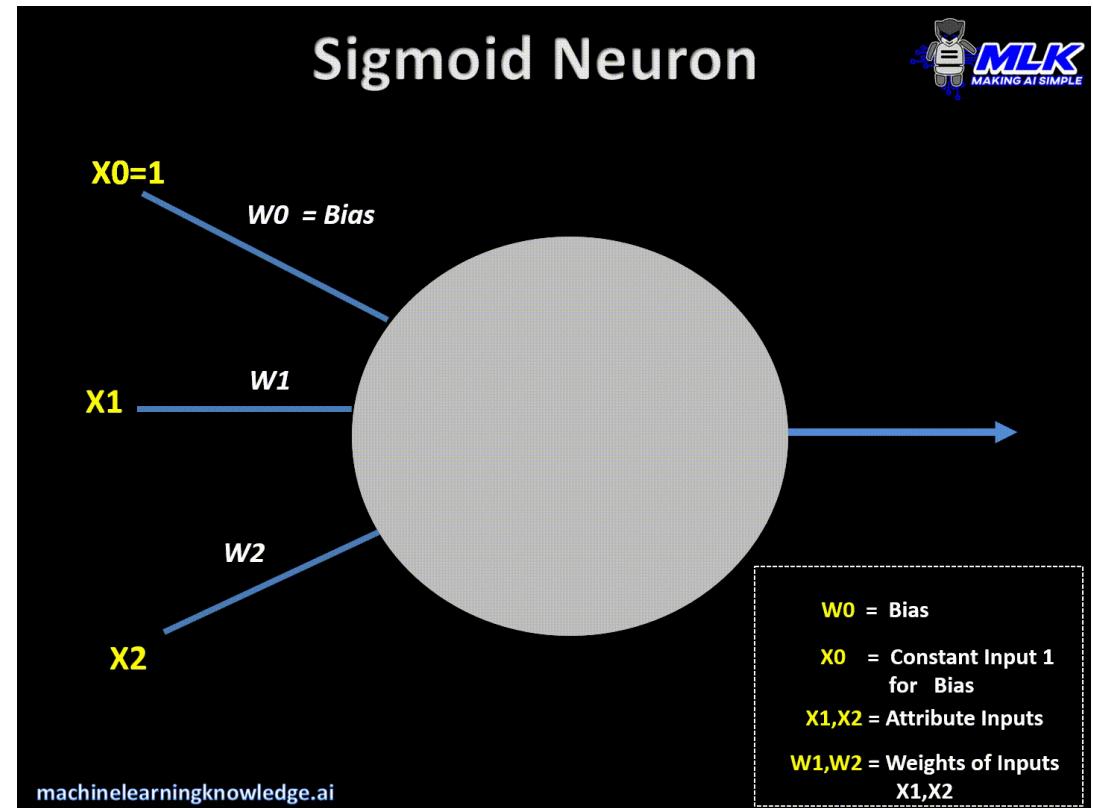
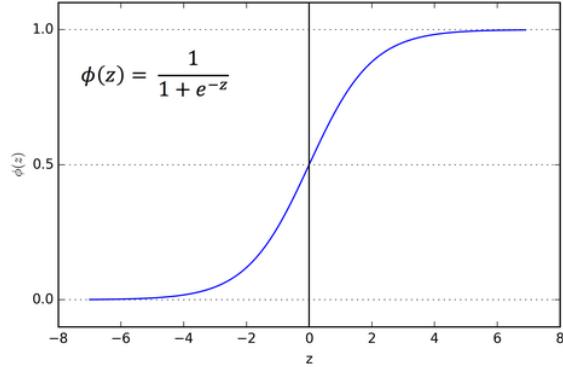
- The most primitive form of artificial neural networks, conceptualized by Frank Rosenblatt in the year 1957, consists of the following parts:
- Inputs: any numerical number
- Weights: Each input has a weight associated with it in the perceptron model. These weights are initially unknown but are learned by Perceptron during the training phase.
- Neuron: A computational unit that has incoming input signals. The input signals are computed, and the output is fired. The neuron further consists of the following two elements :
 1. Summation Function: This simply calculates the sum of incoming inputs multiplied by their respective weights.
 2. Activation Function: Here the activation function is a step function
- Output: binary value 0 or 1.



Binary Step Activation Function

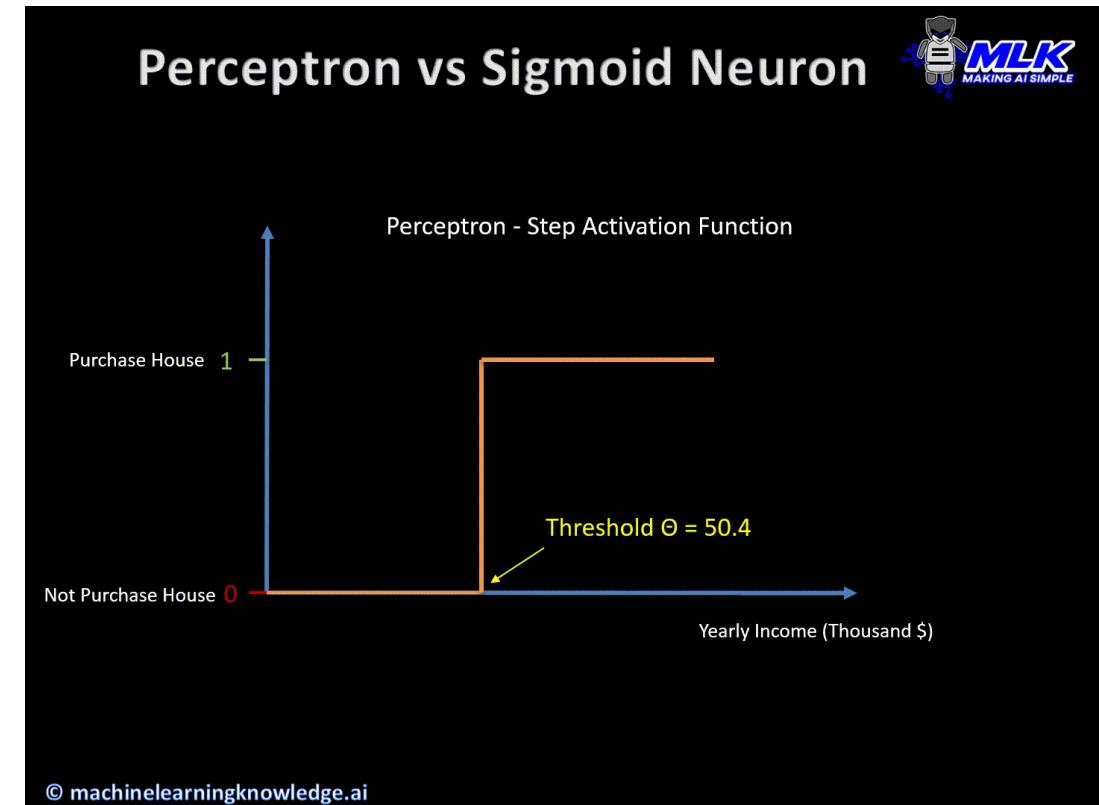
Sigmoid Neuron

- Sigmoid neuron is an artificial neuron that has sigmoid activation function at its core. This is like Perceptron but instead of a step function it has sigmoid function.



Perceptron vs Sigmoid

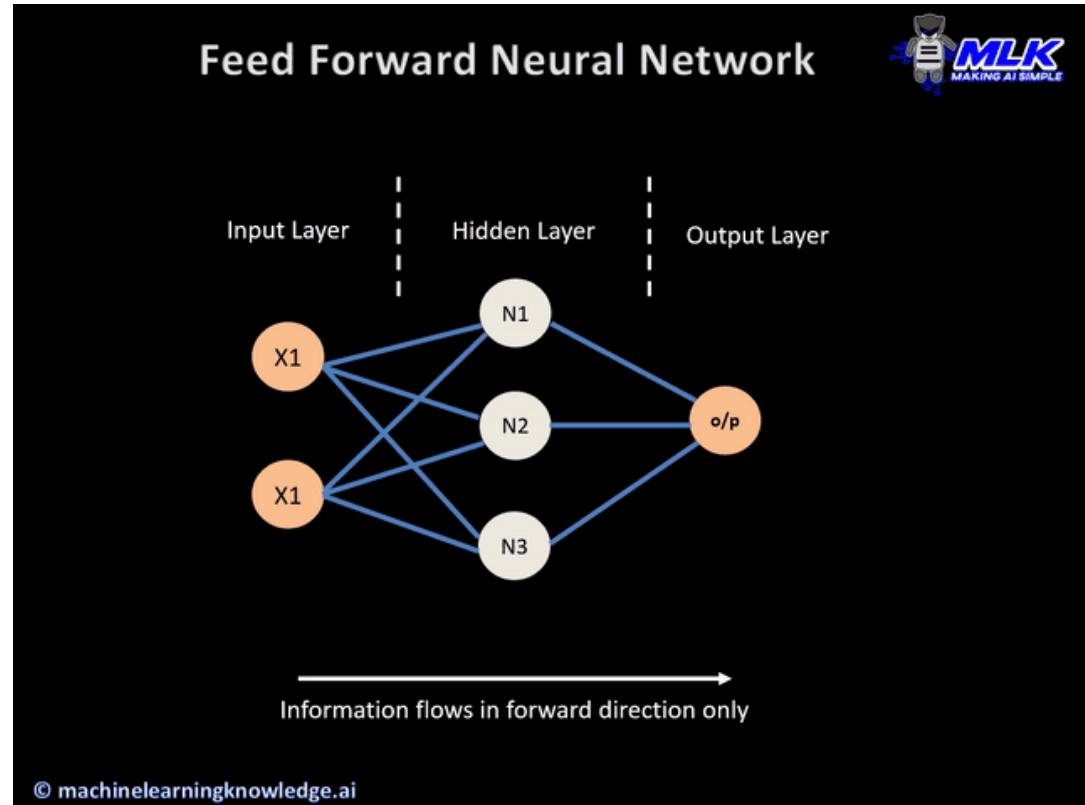
- Perceptron: strict yes or no in the output
- Sigmoid: probability of something in the output



Feed Forward Neural Network



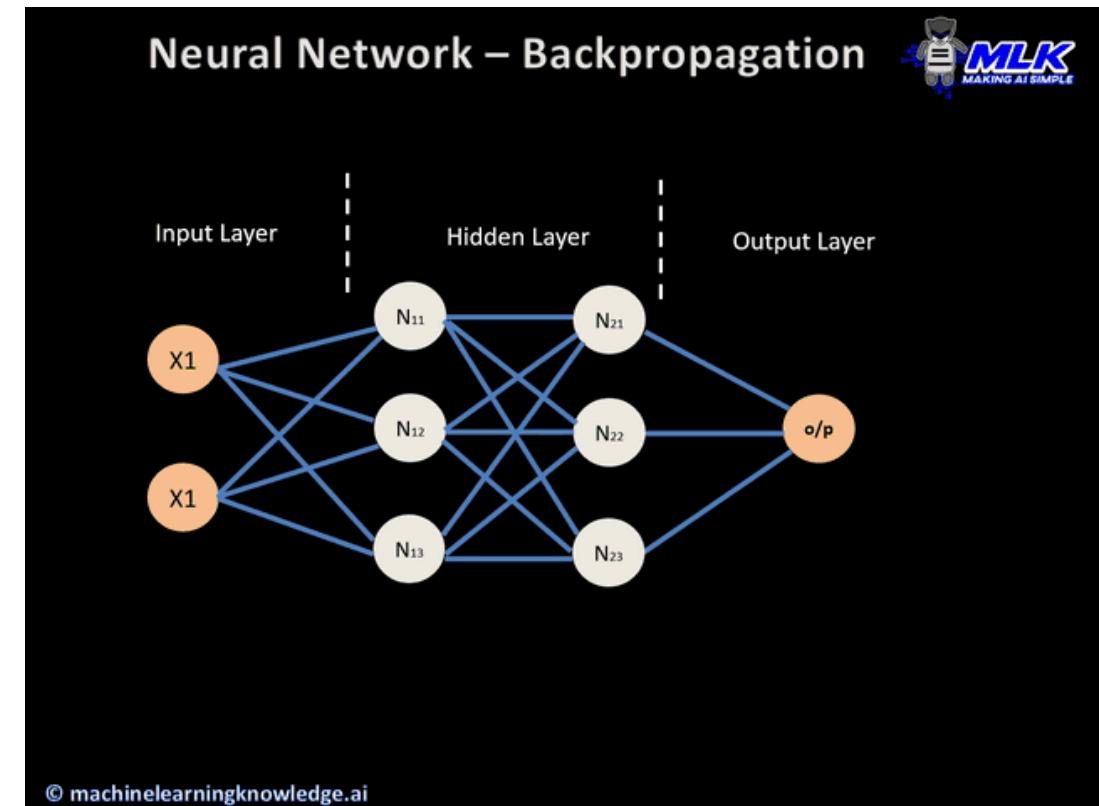
- Formed by stacking multiple neurons together
 - Basically, consists of input layer, hidden layer and output layer
1. Input layer: neurons containing input data, depicted like neurons but not actual artificial neurons, each neuron represents a feature of data
 2. Hidden layer: containing actual artificial neurons
 1. shallow neural network: network with one hidden layer:
 2. Deep neural network: network with many hidden layers
 3. Output layer: representing the network output, The number of output neurons depends on the number of outputs that we are expecting in the problem at hand.
 4. Weights and Bias



© machinelearningknowledge.ai

Backpropagation

- A technique to train the network

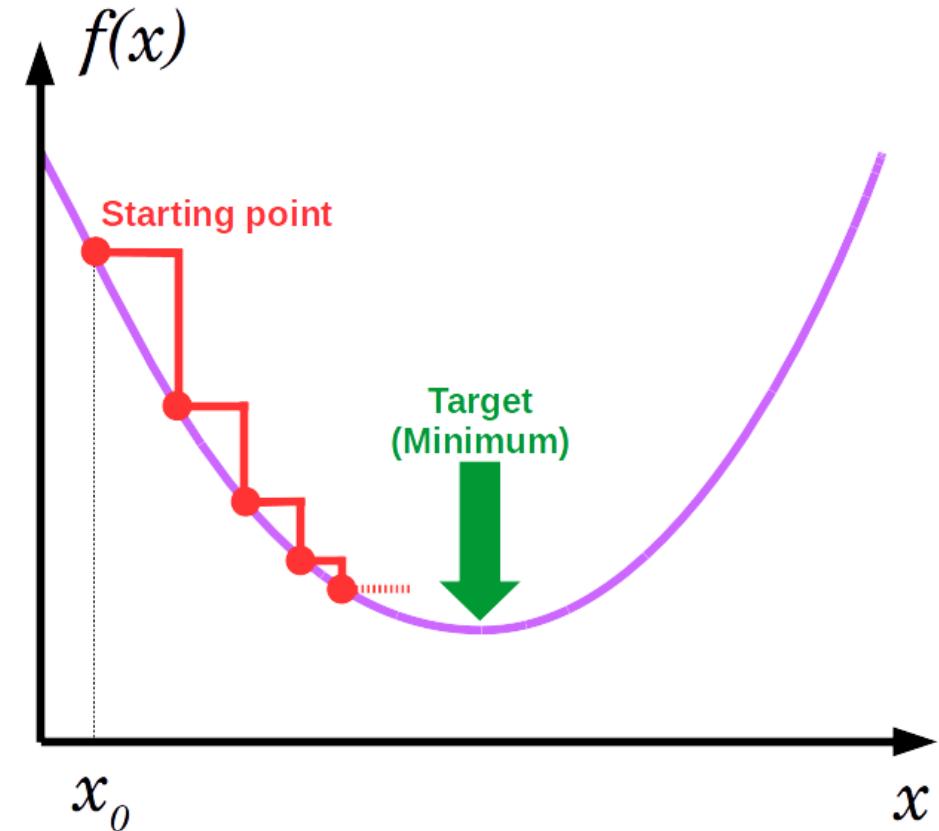


Gradient Descent



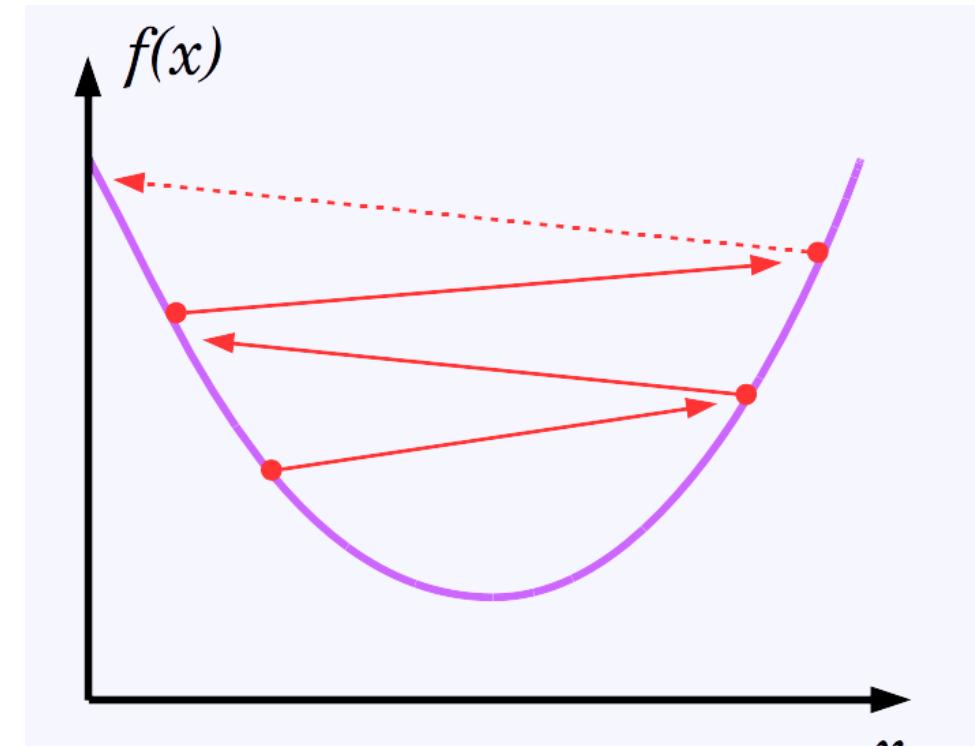
- A large majority of artificial neural networks are based on the gradient descent algorithm.
- Gradient descent is an optimization algorithm for finding the minimum of a function.
- Let's consider the differentiable function $f(x)$ to minimize. The gradient descent algorithm starts at an arbitrary position and iteratively converges to the minimum.

$$x_{n+1} = x_n - \alpha \cdot \frac{df}{dx}(x_n)$$

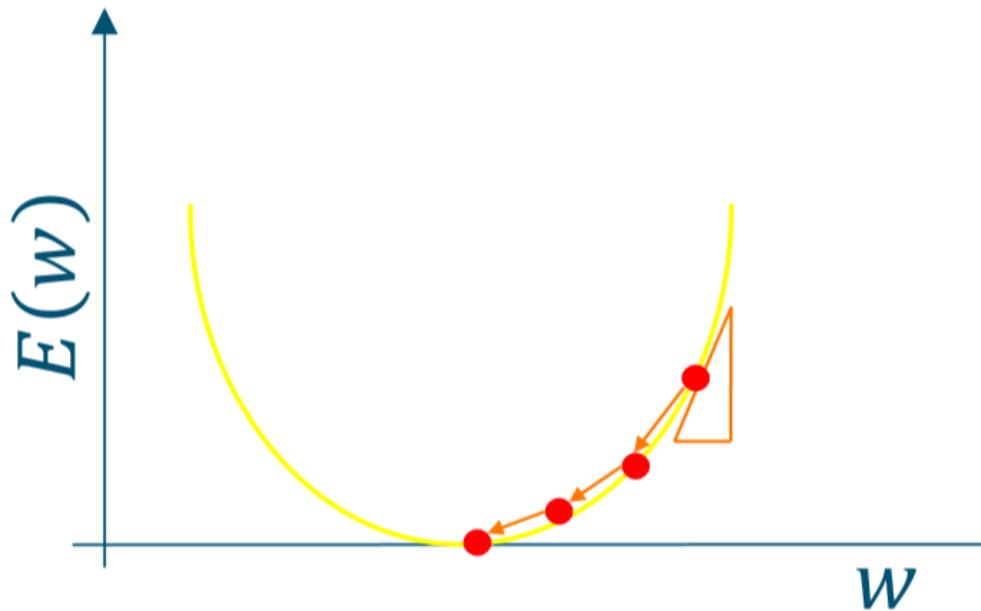


Step size multiplier or Learning rate

- The step size multiplier is a parameter to tune.
- It represents how big will be the step to the next point. The dilemma hidden behind this parameter is that large values will allow fast convergences, but small values will ensure more stability.



Gradient Descent: Weight update



Weight update:

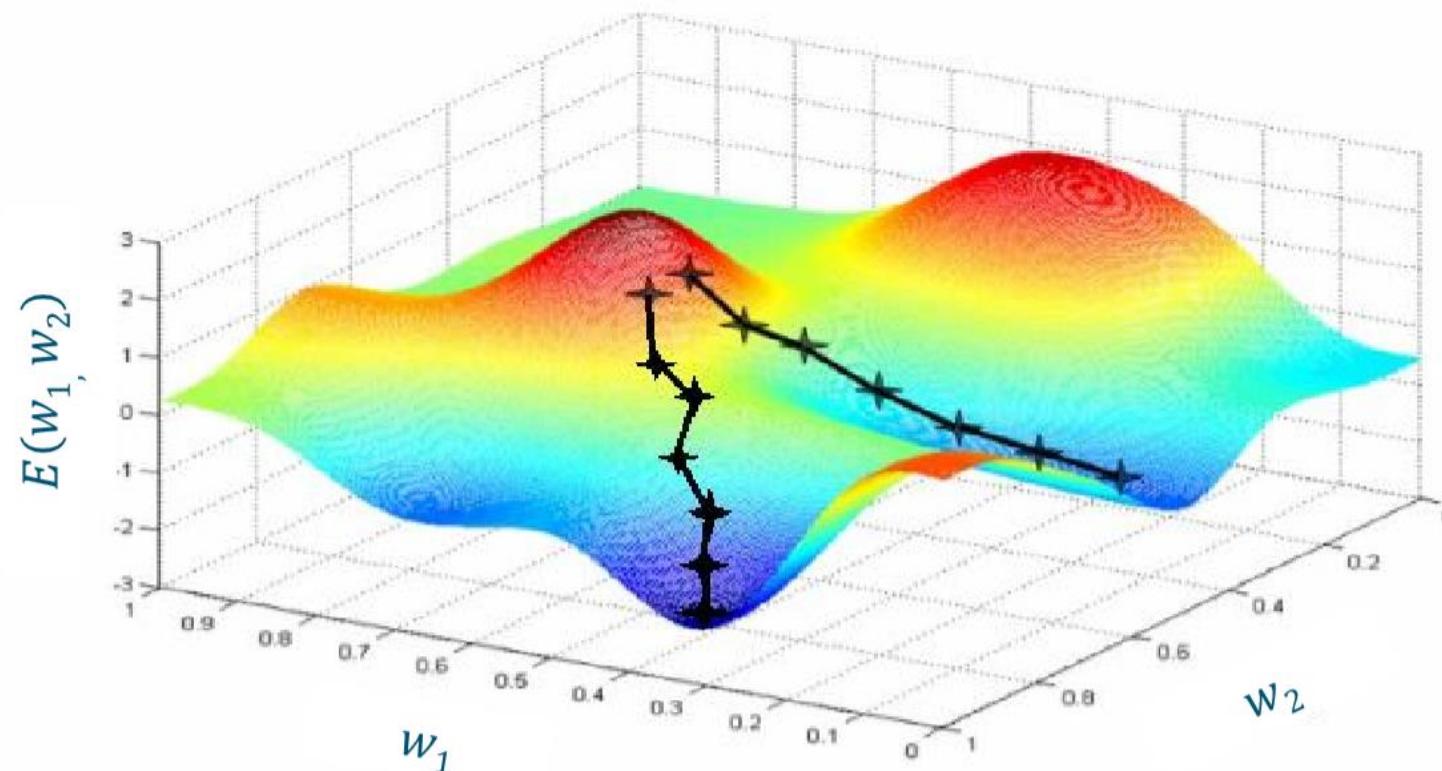
Repeat:

$$w = w + \alpha \times \frac{\partial E}{\partial w}$$

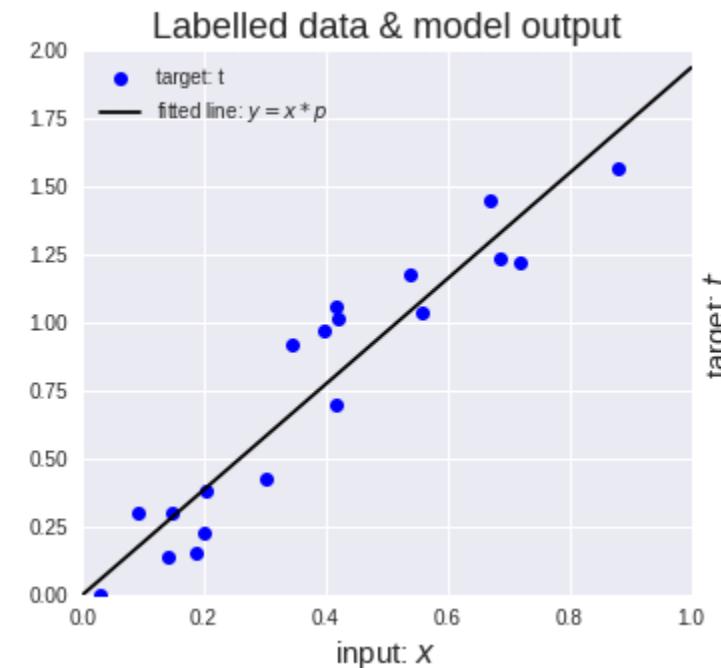
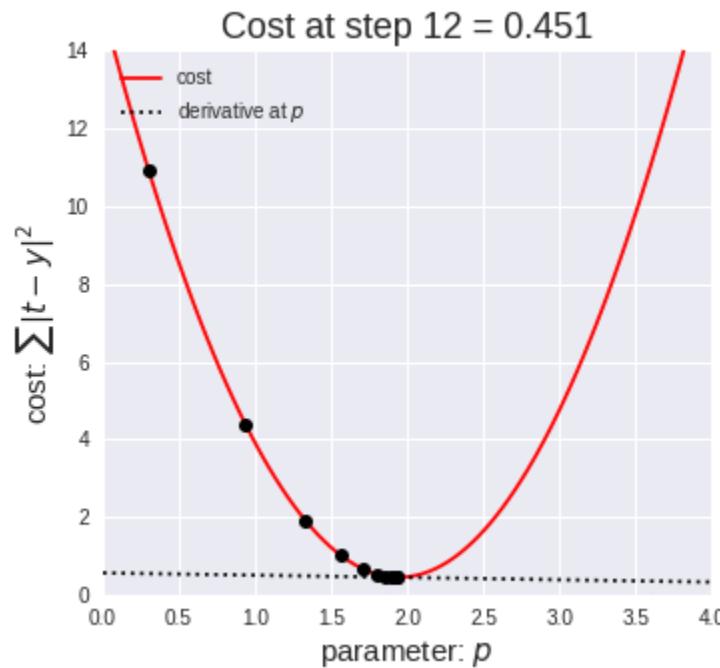
↓ ↓ ↓ ↓ ↓

New weight Old weight Learning Rate Gradient (Direction of descent)

Gradient Descent

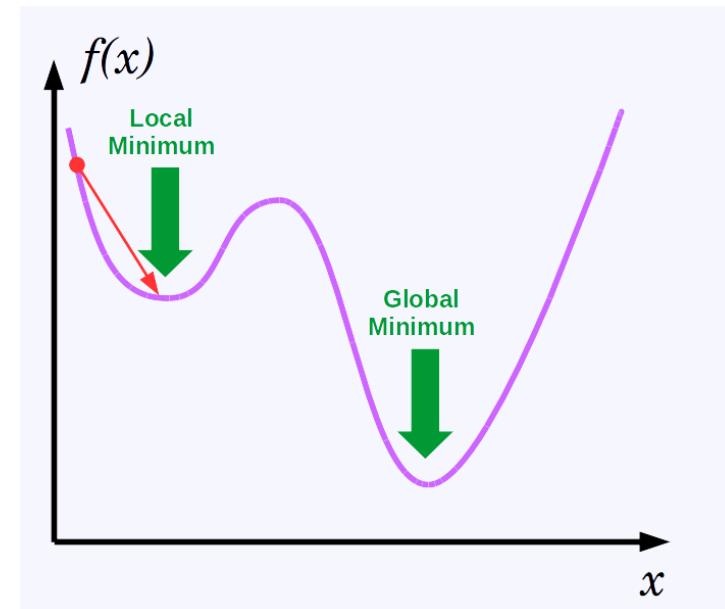


What is happening?



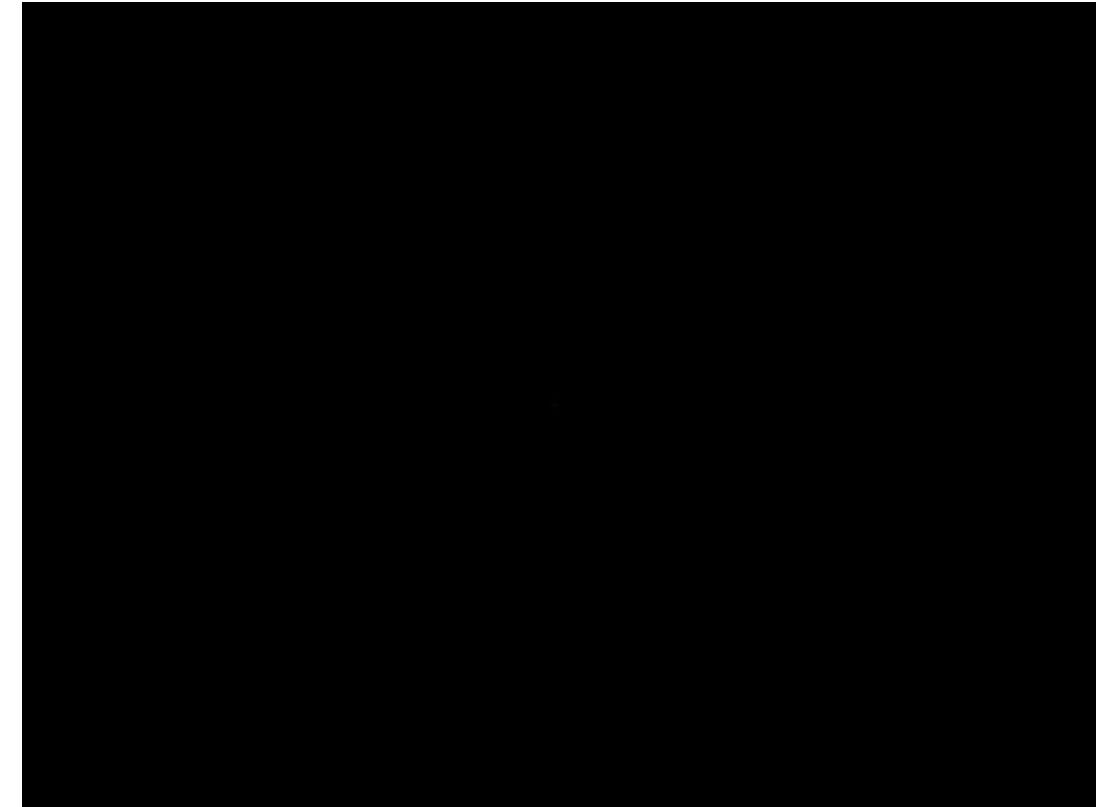
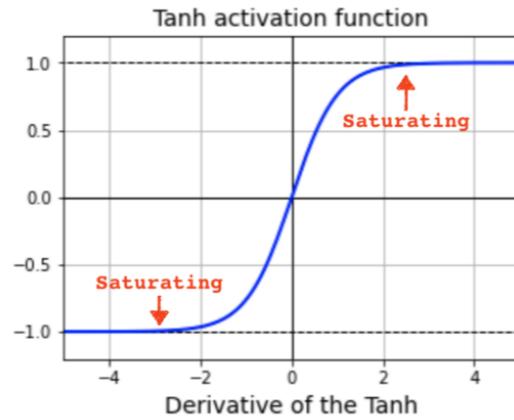
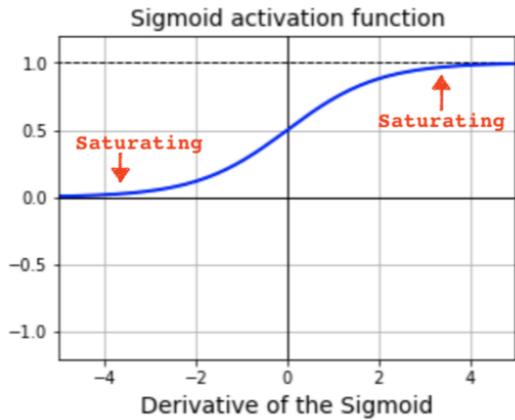
Local minima

- When using the gradient descent algorithm, we must consider the fact that the algorithm can converge to local minima



<https://lucidar.me/en/neural-networks/single-layer-gradient-descent/>

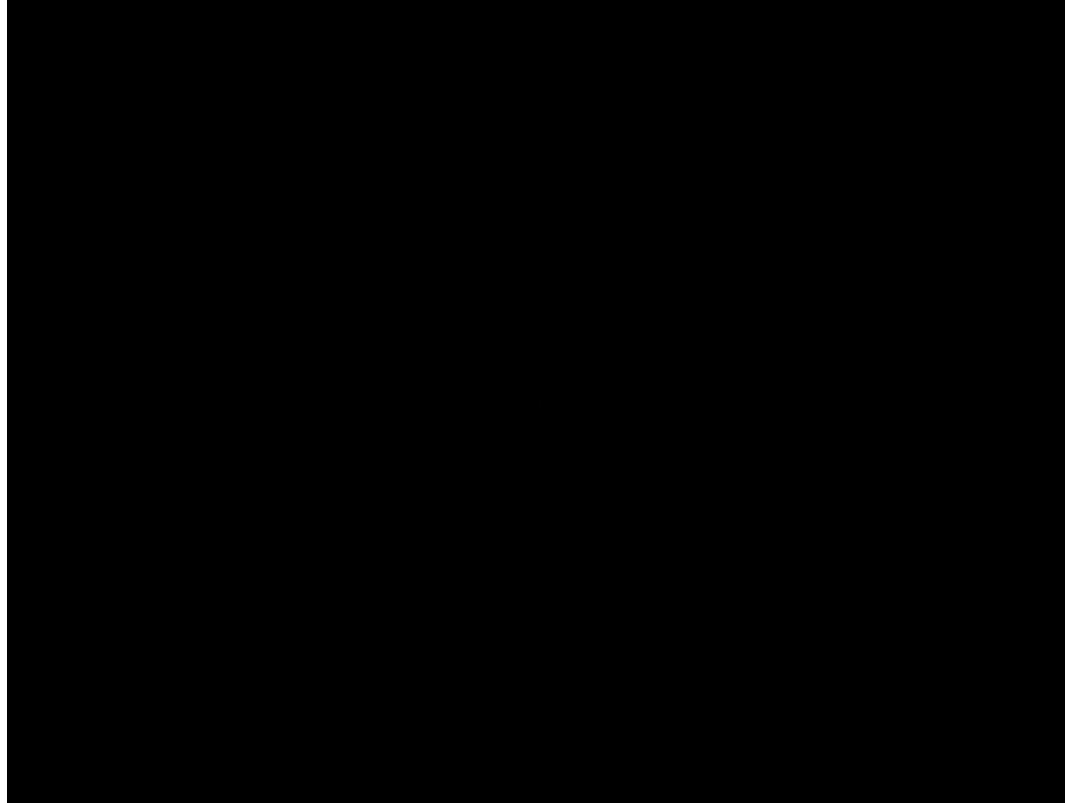
Vanishing Gradient Problem



ReLU for Vanishing Gradients

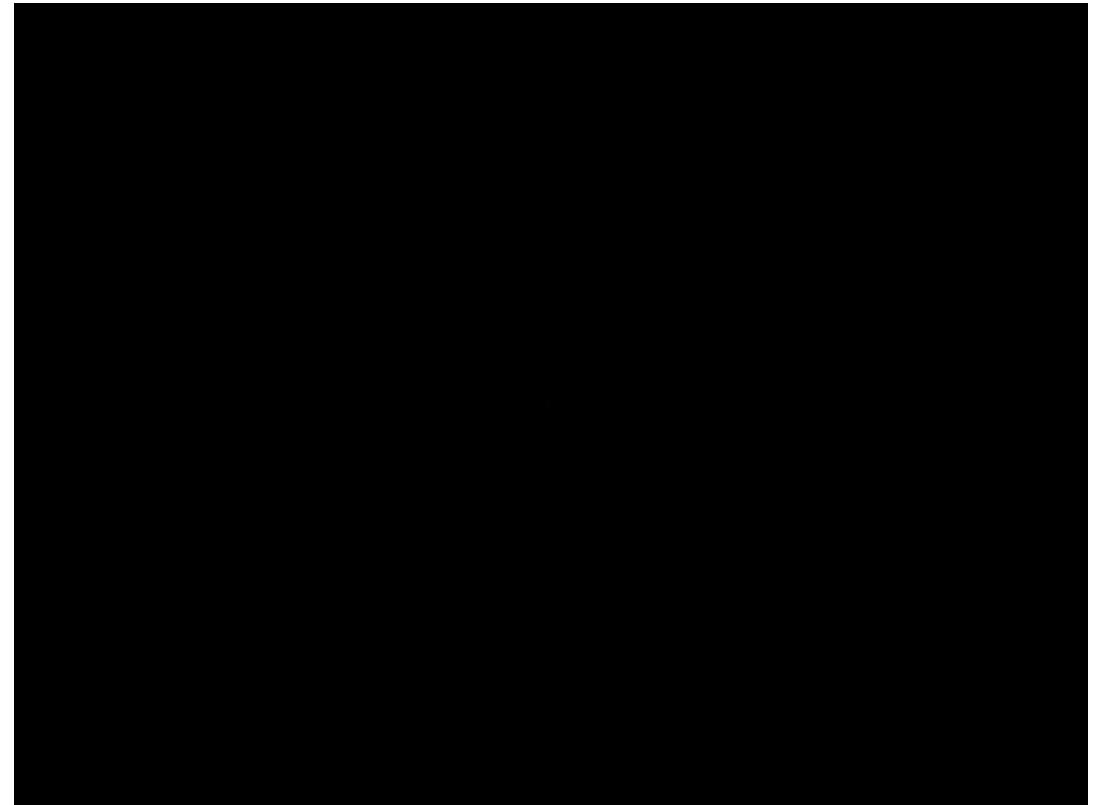


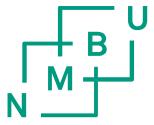
Rectified Linear Activation Function



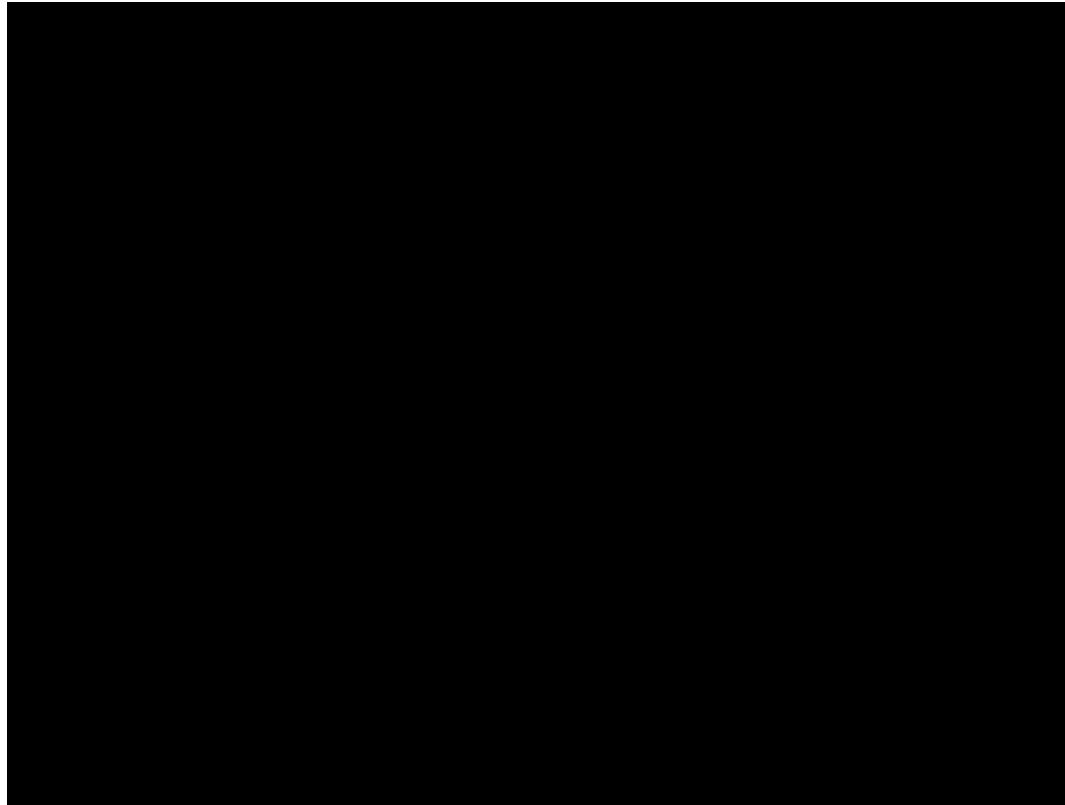
Dying neuron problem and leaky ReLU as solution

- Weights less than zero will not get updated with ReLu
- The node will not have any contribution in the model and has claimed that the neuron is Dead.
- Solution: Use an activation function who slope is never exactly zero



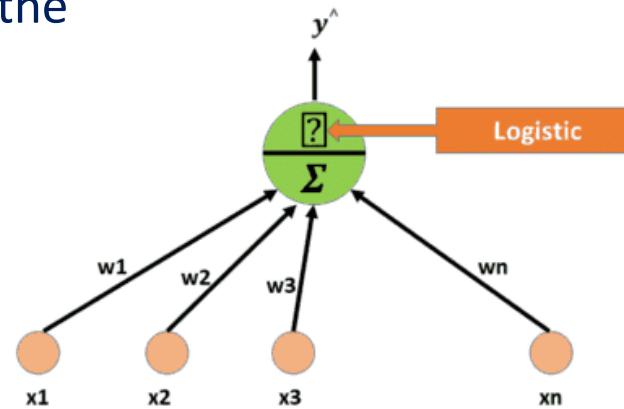


Softmax



Summary

- An activation function is a very important feature of an artificial neural network, they basically decide whether the neuron should be activated or not.

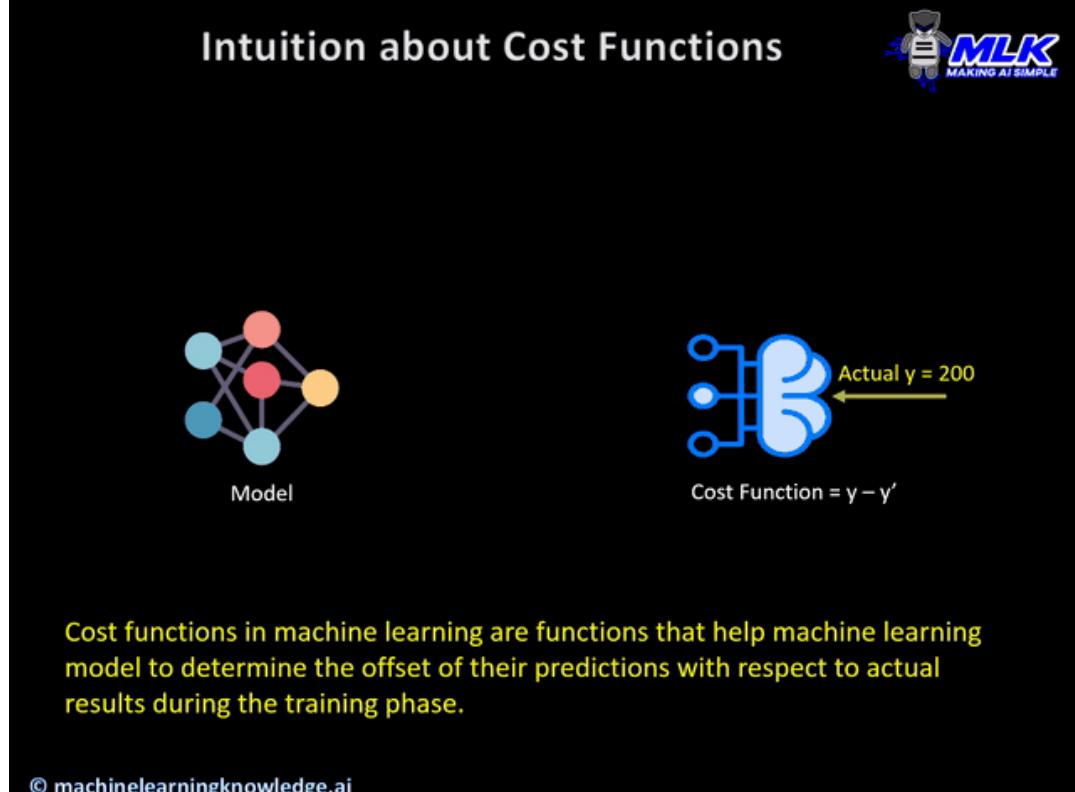


<https://medium.com/@vinodhb95/lost-function-and-its-type-f7fcec45c724>

Loss function or Cost function

A function that takes both predicted outputs by the model and actual outputs and calculates how much wrong the model was in its prediction.

Intuition about Cost Functions



The diagram illustrates the concept of a cost function in machine learning. On the left, a neural network structure is labeled "Model", showing multiple layers of nodes with connections. On the right, a blue figure representing a machine learning model is shown with an arrow pointing to the text "Actual $y = 200$ ". Below this, the formula "Cost Function = $y - y'$ " is displayed. The background is black, and the overall theme is educational, as indicated by the "MLK MAKING AI SIMPLE" logo in the top right corner.

Model

Actual $y = 200$

Cost Function = $y - y'$

Cost functions in machine learning are functions that help machine learning model to determine the offset of their predictions with respect to actual results during the training phase.

© machinelearningknowledge.ai

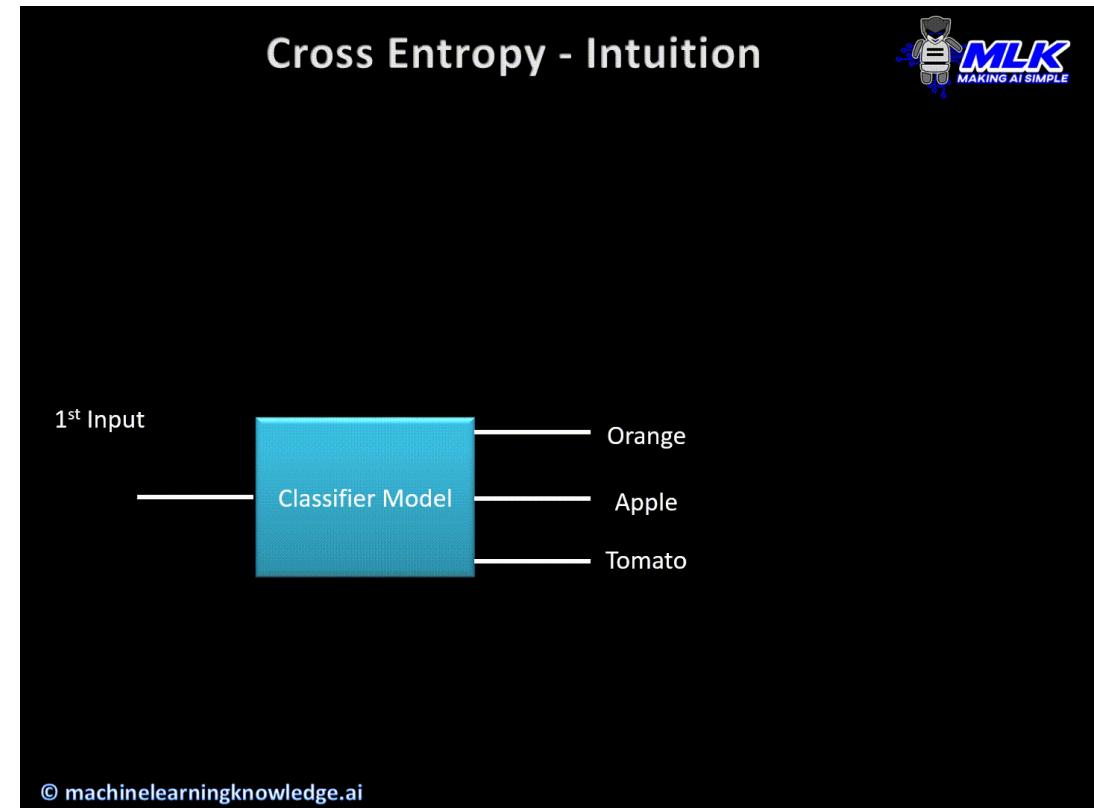
Regression cost function

Mean Squared Error (MSE) or Mean Absolute Error (MAE)?



Cost functions for Classification problems

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.





Epoch

- **Epoch refers to one cycle through the full training dataset.**
- Usually, training a neural network takes more than a few epochs.
- Increasing the number of epochs doesn't always mean that the network will give better results.
- So basically, by trial-and-error, we choose several epochs at which the results are still the same after a very few cycles.



Iteration and batch

- **Batch is the number of training samples or examples in one iteration.** The higher the batch size, the more memory space we need.
- For each complete epoch, we have several iterations. **Iteration is the number of batches or steps through partitioned packets of the training data, needed to complete one epoch.**

How to apply for images???

Let's start 'simple'

Input Image



Output Class

Healthy?

Damaged?

Why finding patterns in raw data is difficult

- **We See a bad Orange**



- **The Machine “sees” This**

```
1010011000101010111111
0010111010001100111000
1111100100110110010010
10111001010101010111
1001111000101100011110
0001100110101001100010
101011111001011101000
1100111000111110010011
0110010010101110010101
0101010111100111100010
```

Manual Feature Extraction

- Pretend for a moment that this orange variety should be as orange as possible, and be at least a certain size and smooth texture.

Real World



Raw Data

```
1010011000101010111111
0010111010001100111000
1111100100110110010010
101110010101010101111
1001111000101100011110
0001100110101001100010
101011111001011101000
1100111000111110010011
0110010010101110010101
0101010111100111100010
```

Features

Feature
extraction

Orange color
percentage

Texture

With the extracted features, decision making is a lot easier

Generic “Shallow” Machine Learning

Raw Data

1010011000101010111111001

0111010001100111000111110

01001101100100101

10101010101111001

1100011110000110011010100

110001010101111100101110

Features

f0, f1, f2
f3, f4, f5
f6, f7, f8

Feature Extraction

CLASSIFIER

ACCEPT
or
REJECT

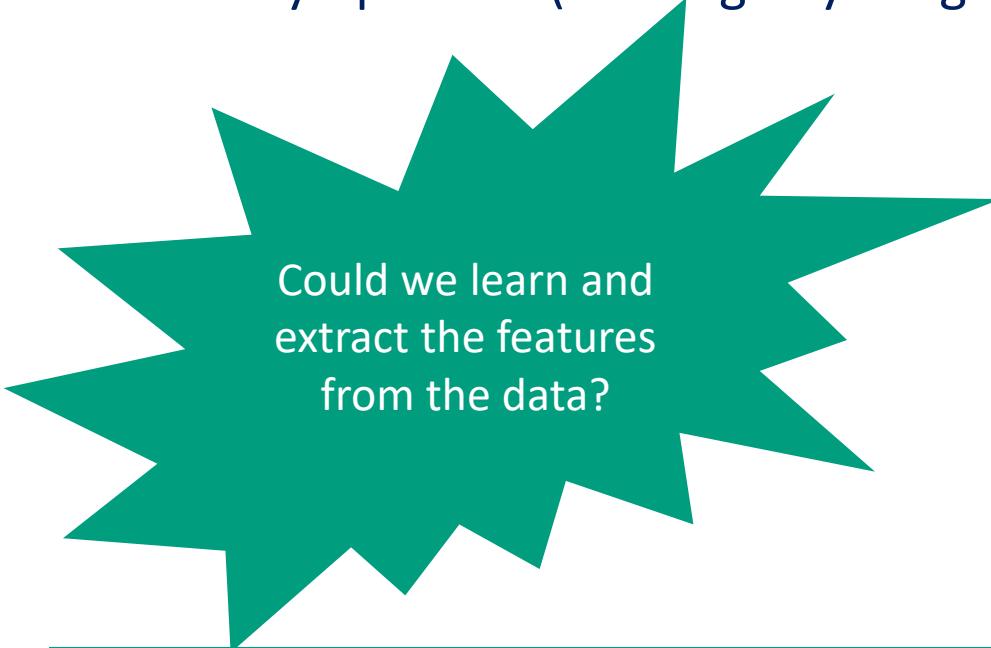
- Features are often manually chosen (expert knowledge?)
 - The classifier is trainable, but typically makes decisions based directly on the extracted features.
- So the features had better be informative!

Problems with Shallow Learning

¶ How do we choose our features?

¶ Expert knowledge can help, but:

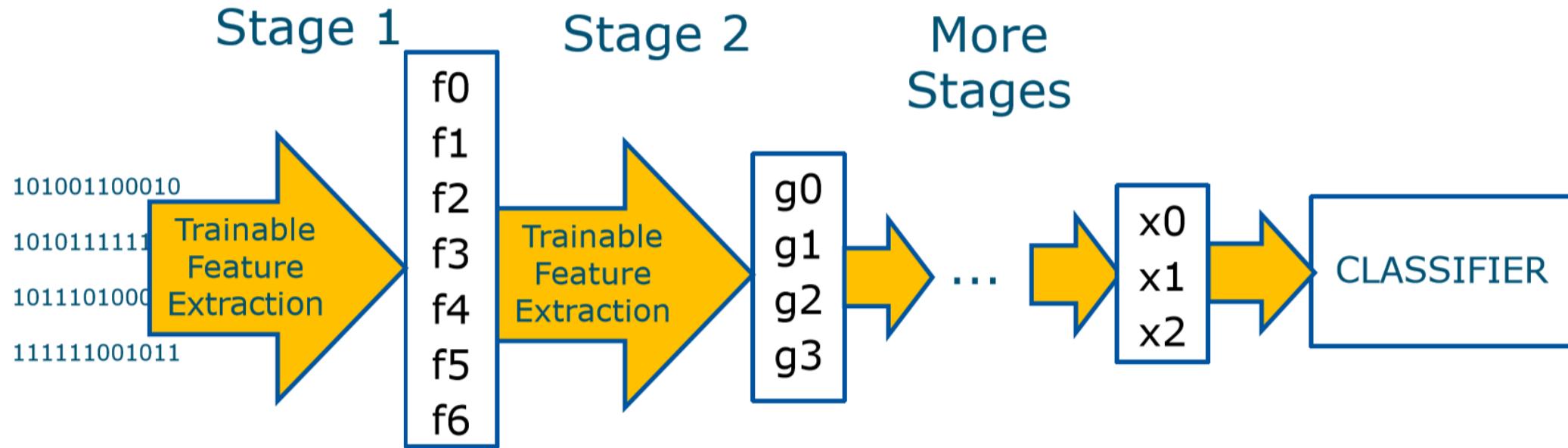
- Are the features robust?
- Are they optimal? (Missing anything non-obvious?)



Could we learn and extract the features from the data?

Certain approaches train one step of feature extraction, but the jump from raw data to high-level information is quite high...

From Shallow to Deep Learning

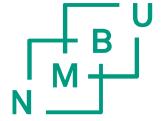


- Recent progress now allows us to train many stages of feature extraction rapidly without manual tinkering.
- Each stage can make it easier for subsequent stages to extract useful information.
- At the classifier stage, features are highly informative.

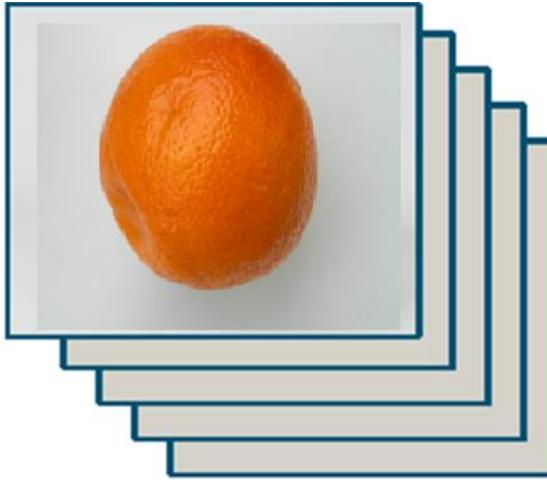


Nice idea, how do we do this?

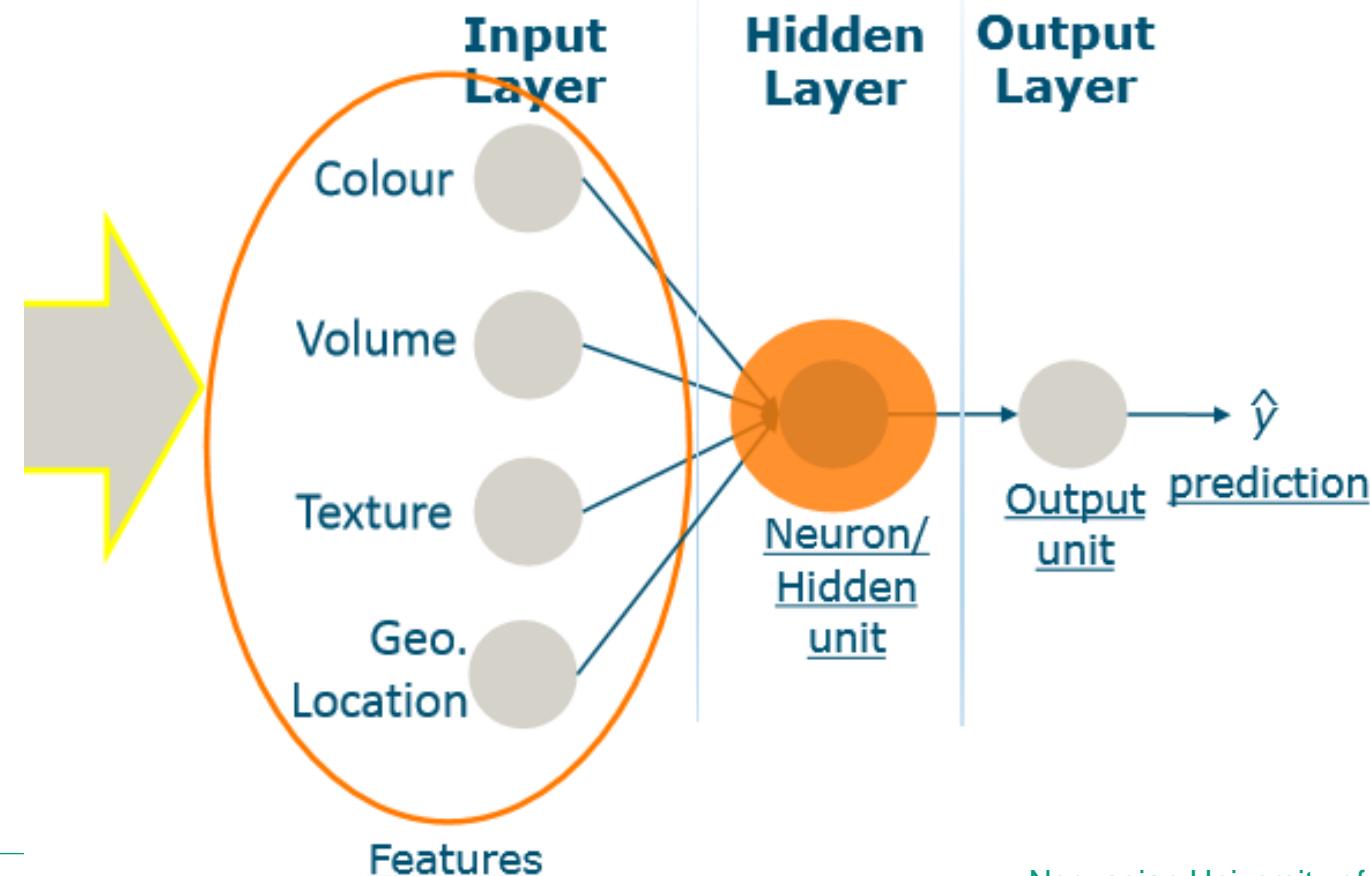
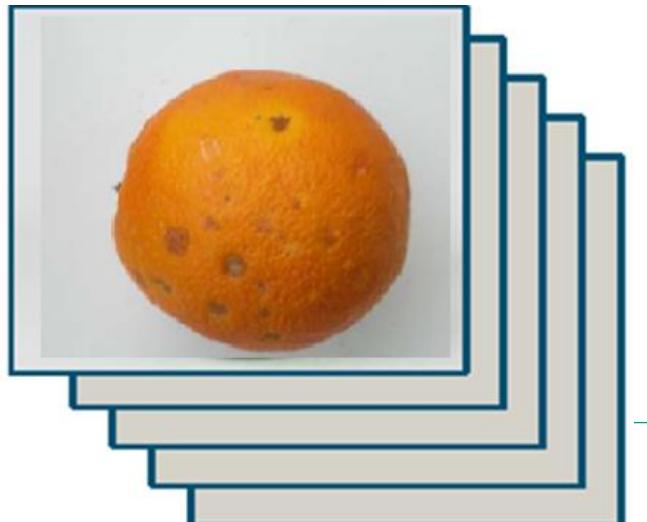
Neural Networks



Positive examples



Negative examples





Deep Networks

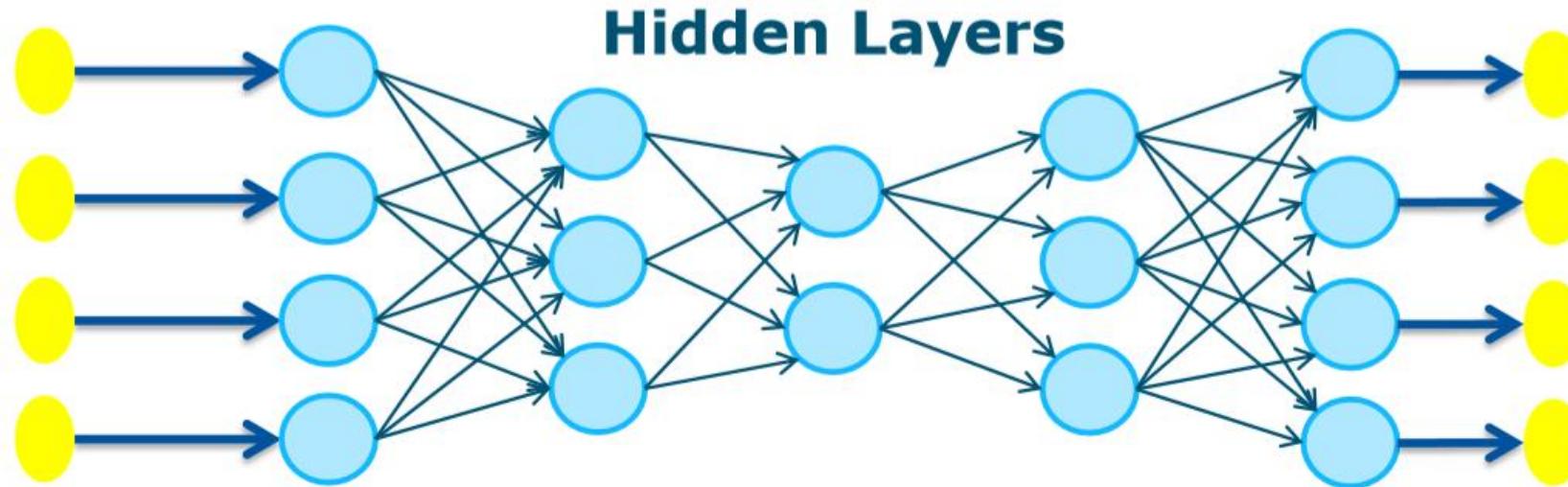
Challenges

The basic concepts of neural networks have been known for decades ...

... but until recently progress was slow.

Why?

Deep Networks



To solve challenging problems, we often need to add many neurons per hidden layer, and also many hidden layers.



Problems With Deep Networks

- Gradients become tiny as the network gets deeper
- Steps are small, so training is slow
- Without enough data, large networks tend to REMEMBER instead of UNDERSTAND
(technically: OVERFITTING vs. GOOD GENERALIZATION)

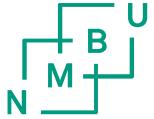
Not enough raw computational power



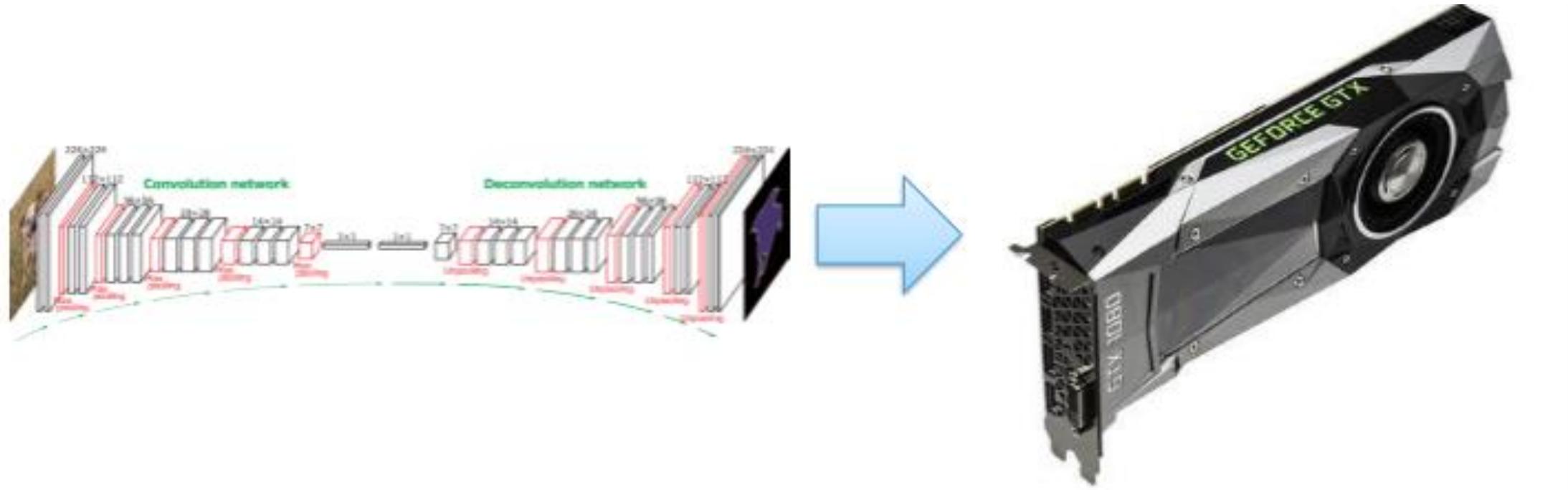
Why is deep learning successful now?

- Better network design choices (for example, convolutional nets)
- Better approaches to training a network Massive amounts of data (for certain tasks)
- Computing hardware much more powerful and better suited to the task (GPUs)

Efficient Implementation

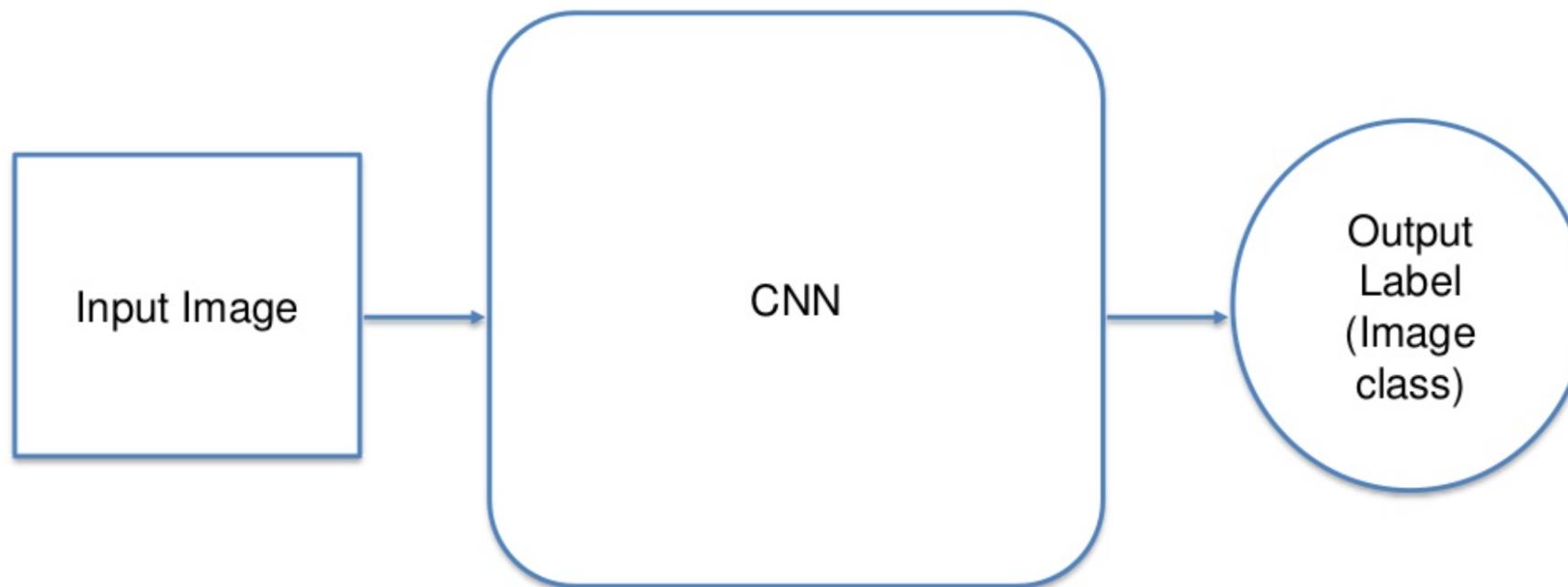


- Libraries such as PyTorch, Caffe, TensorFlow and Theano can perform neural network computations on GPUs.

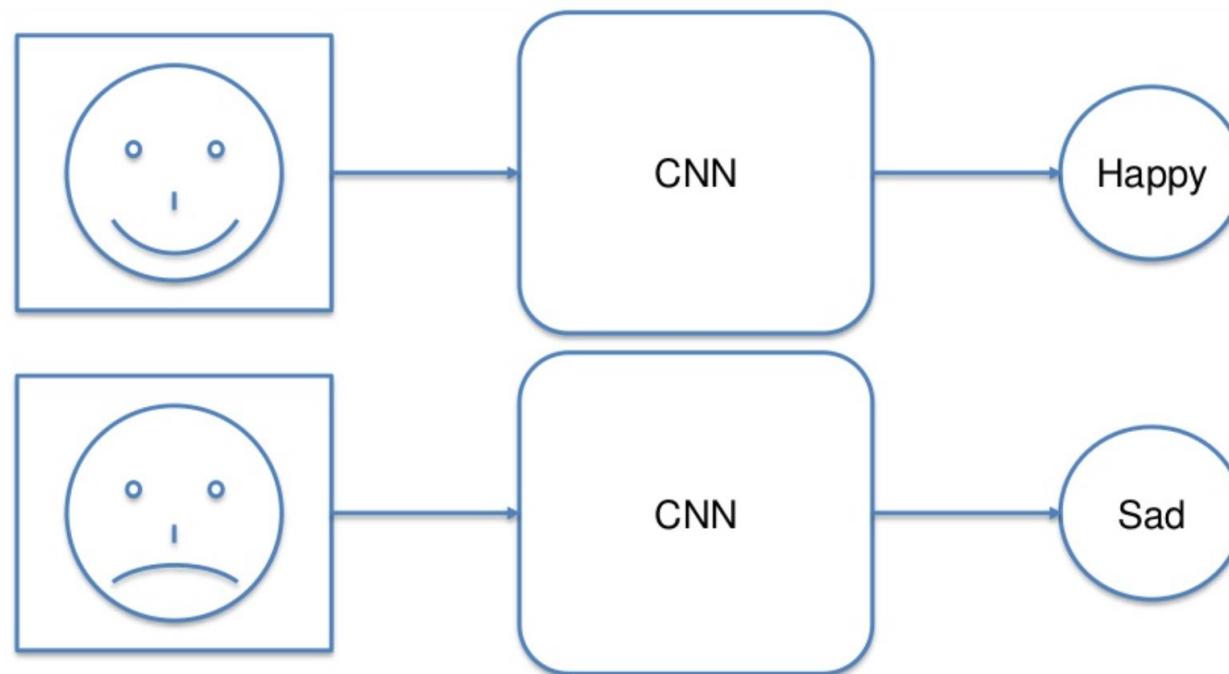


Huge speed increases! Essential to working with big datasets.

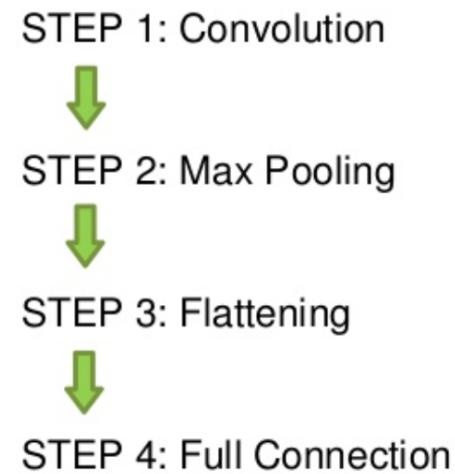
Convolutional Neural Network



Convolutional Neural Network

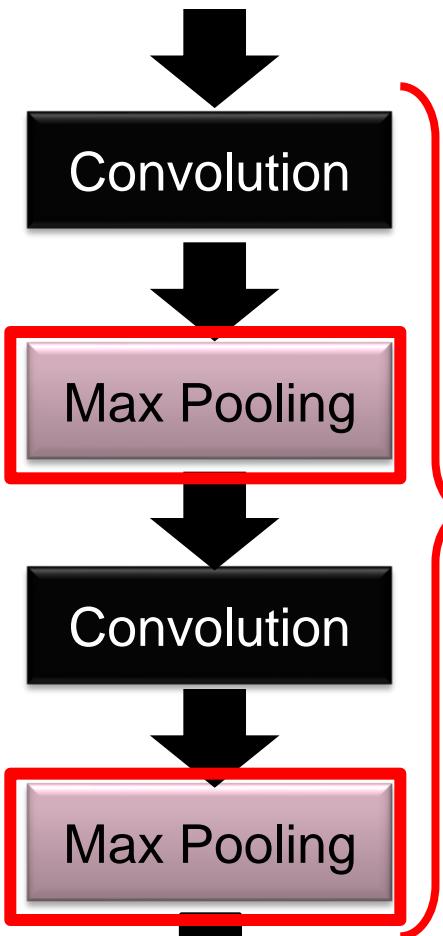
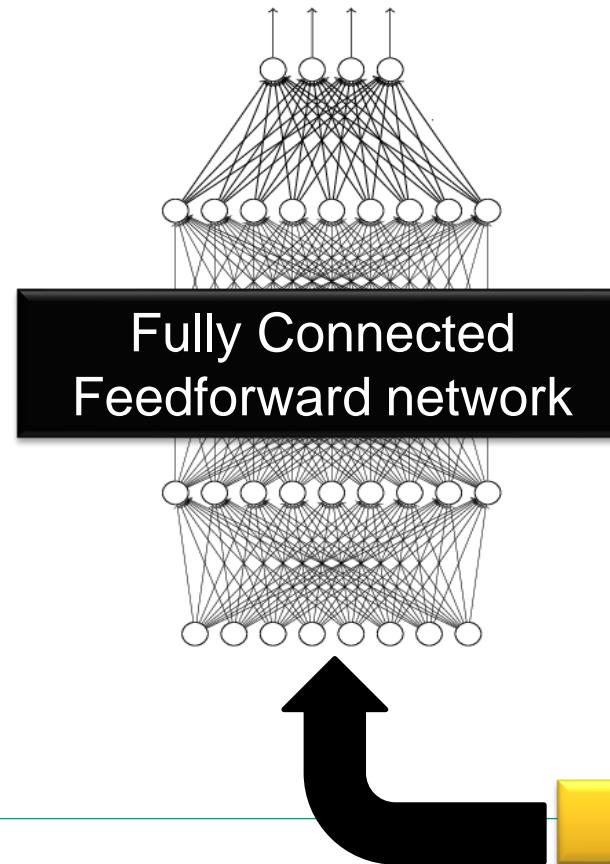


Convolutional Neural Network



The whole CNN

cat dog



Can
repeat
many
times



Step 1: Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|--|--|--|--|
| 0 | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|--|--|--|
| 0 | 1 | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|--|--|
| 0 | 1 | 0 | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|--|
| 0 | 1 | 0 | 0 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | | | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | | | |
| | | | | |
| | | | | |
| | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | | |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | |
| | | | | |
| | | | | |
| | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector

=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| | | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | | | | |
| | | | | |
| | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

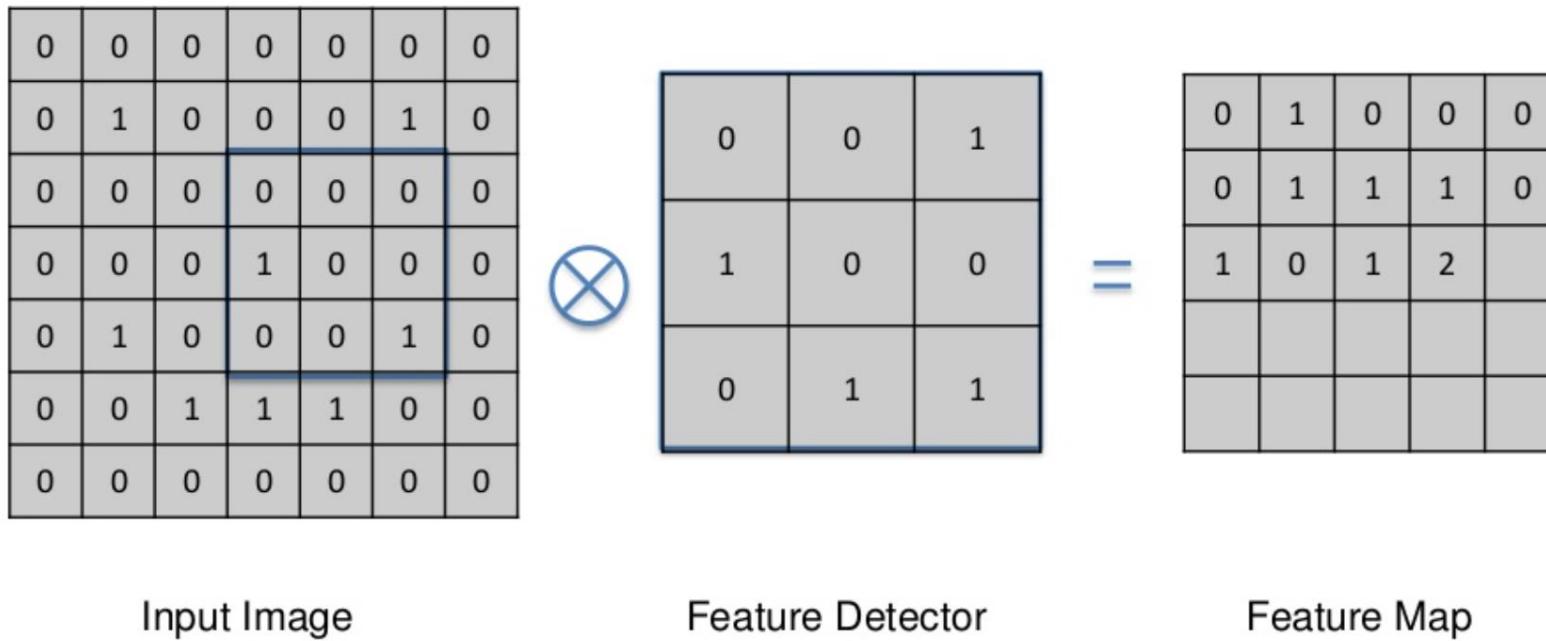
Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | | |
| | | | | |
| | | | | |

Feature Map

Step 1: Convolution



Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

 $=$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| | | | | |
| | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | | | | |

Feature Map

N M B U

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | | | |
| | | | | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

 $=$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | |
| | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

 $=$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

A blue equals sign symbol.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

 $=$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

 $=$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | | |

Input Image

Feature Detector

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Input Image

Feature Detector

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | |

Feature Map

Step 1: Convolution

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Input Image

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

Feature Detector



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

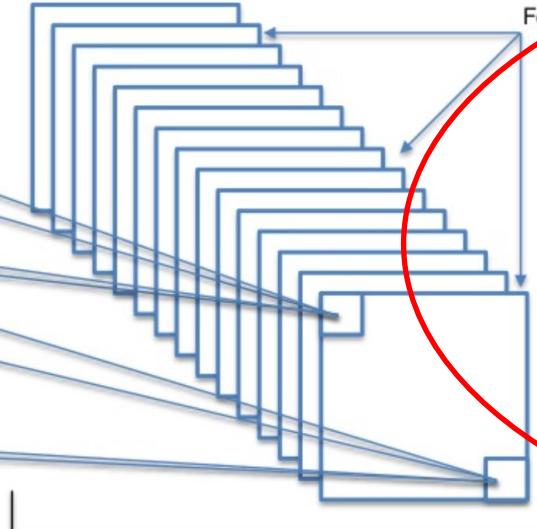
Feature Map

Step 1: ReLU Layer

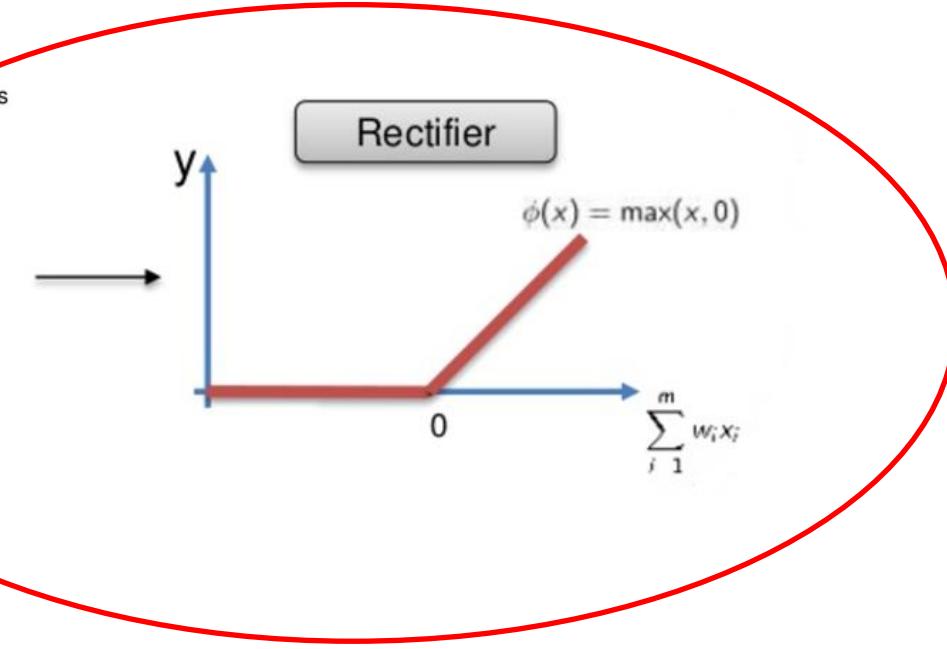
We create many feature maps to obtain our first convolution layer

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



Feature Maps



Number of channels depends on number of filters

Max Pooling



Why pooling?

- Subsampling pixels will not change the object

bird



Subsampling

bird



We can subsample the pixels to make image smaller
fewer parameters to characterize the image

N M B U

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



| | | |
|---|--|--|
| 1 | | |
| | | |
| | | |

Pooled Feature Map

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



| | | |
|---|---|--|
| 1 | 1 | |
| | | |
| | | |

Pooled Feature Map

Max pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling

| | | |
|---|---|---|
| 1 | 1 | 0 |
| | | |
| | | |

Pooled Feature Map

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | | |
| | | |

Pooled Feature Map



Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | |
| | | |

Pooled Feature Map

N M B U

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Max Pooling

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| | | |

Feature Map

Pooled Feature Map

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | | |

Pooled Feature Map

N M B U

Max pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | 2 | |

Pooled Feature Map

Max Pooling

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 2 | 1 |
| 1 | 4 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 1 |

Feature Map

Max Pooling



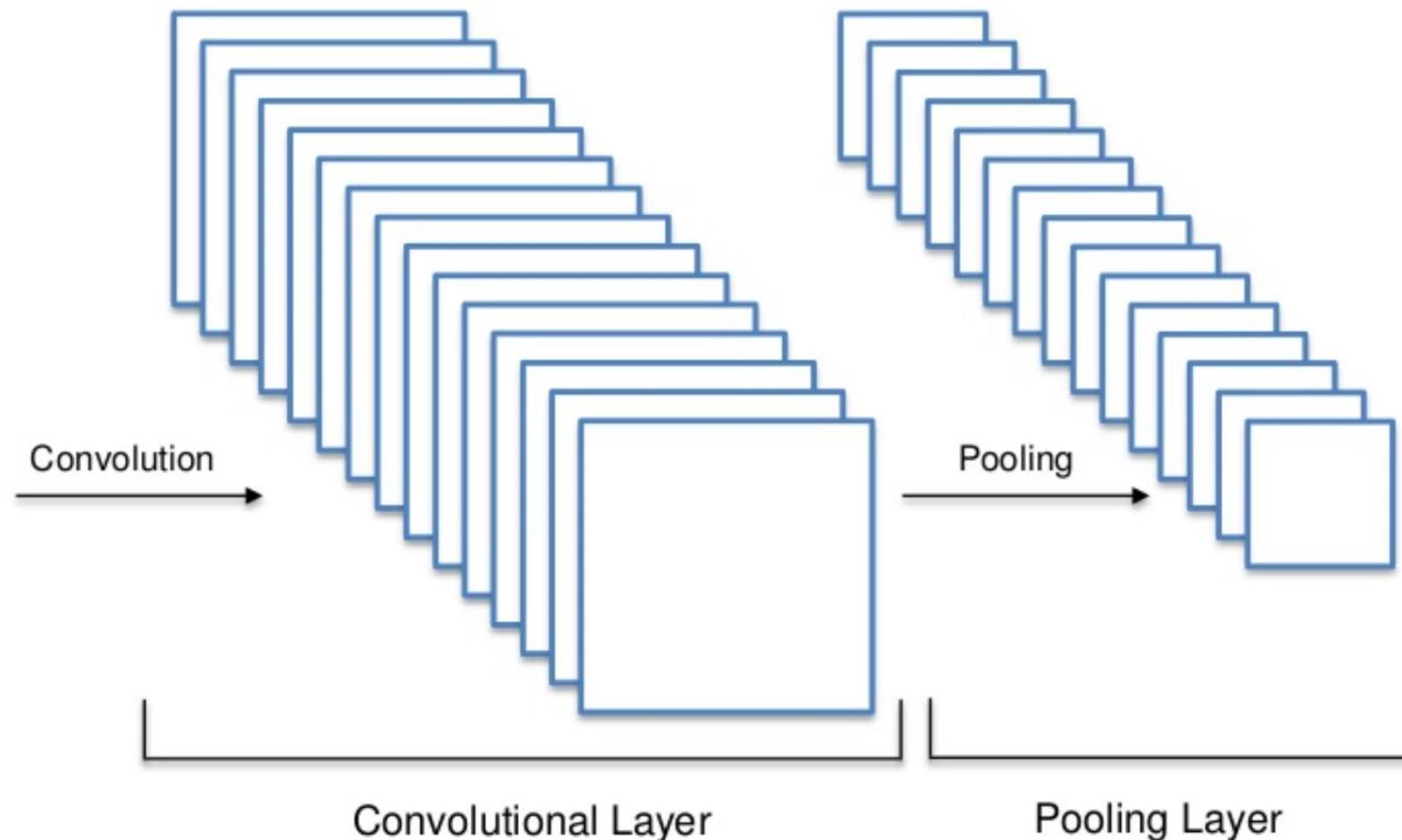
| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | 2 | 1 |

Pooled Feature Map

Max Pooling

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input Image



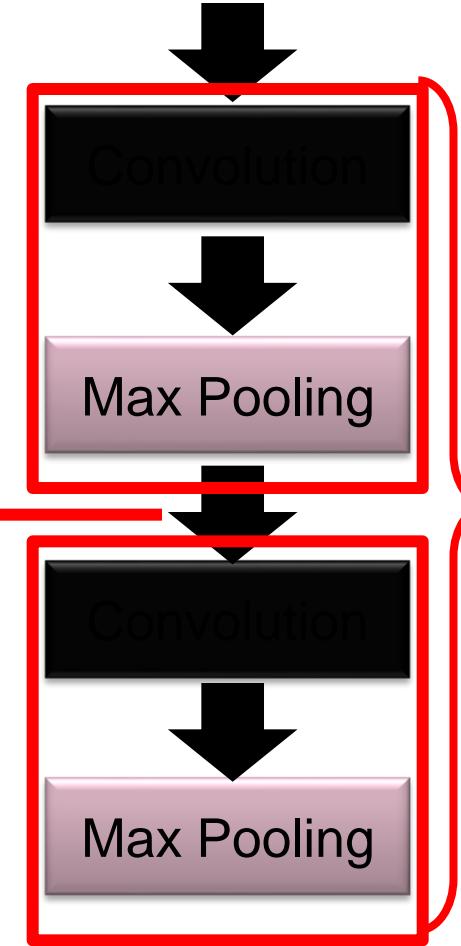
The whole CNN

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | 2 | 1 |

A new image

Smaller than the original image

The number of channels is the number of filters



Example



Flattening

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | 2 | 1 |

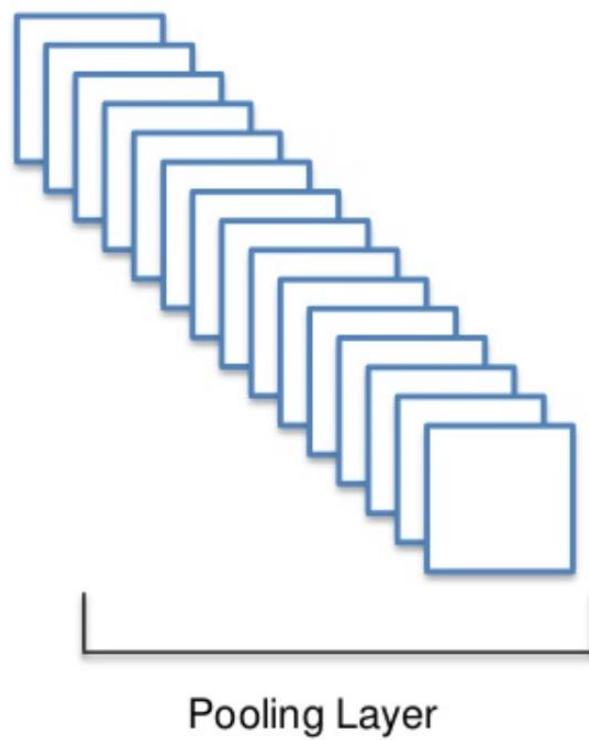
Pooled Feature Map

Flattening 

| |
|---|
| 1 |
| 1 |
| 0 |
| 4 |
| 2 |
| 1 |
| 0 |
| 2 |
| 1 |

Flattennig

N M B U



Flattening

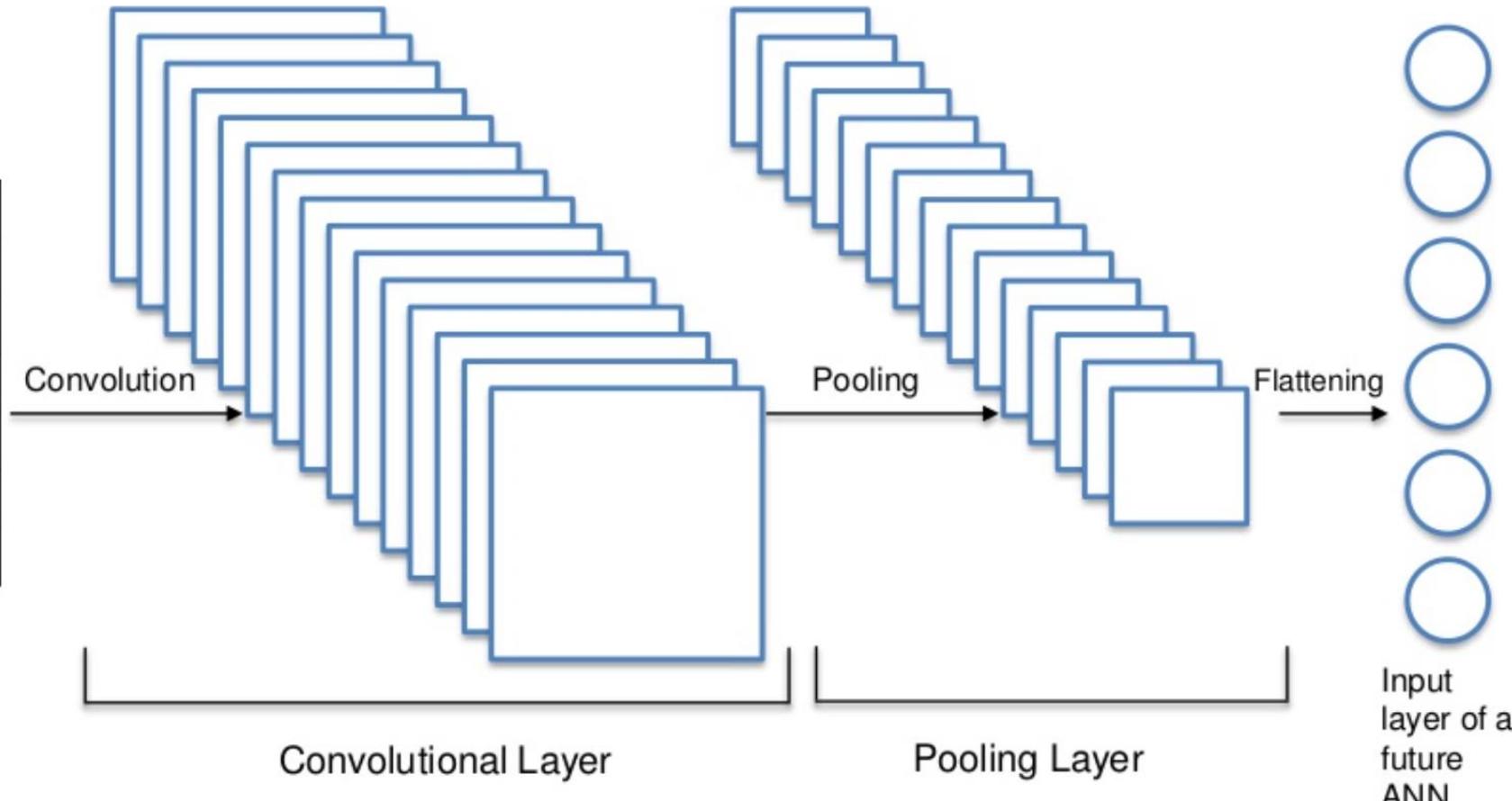


Input layer of a future ANN

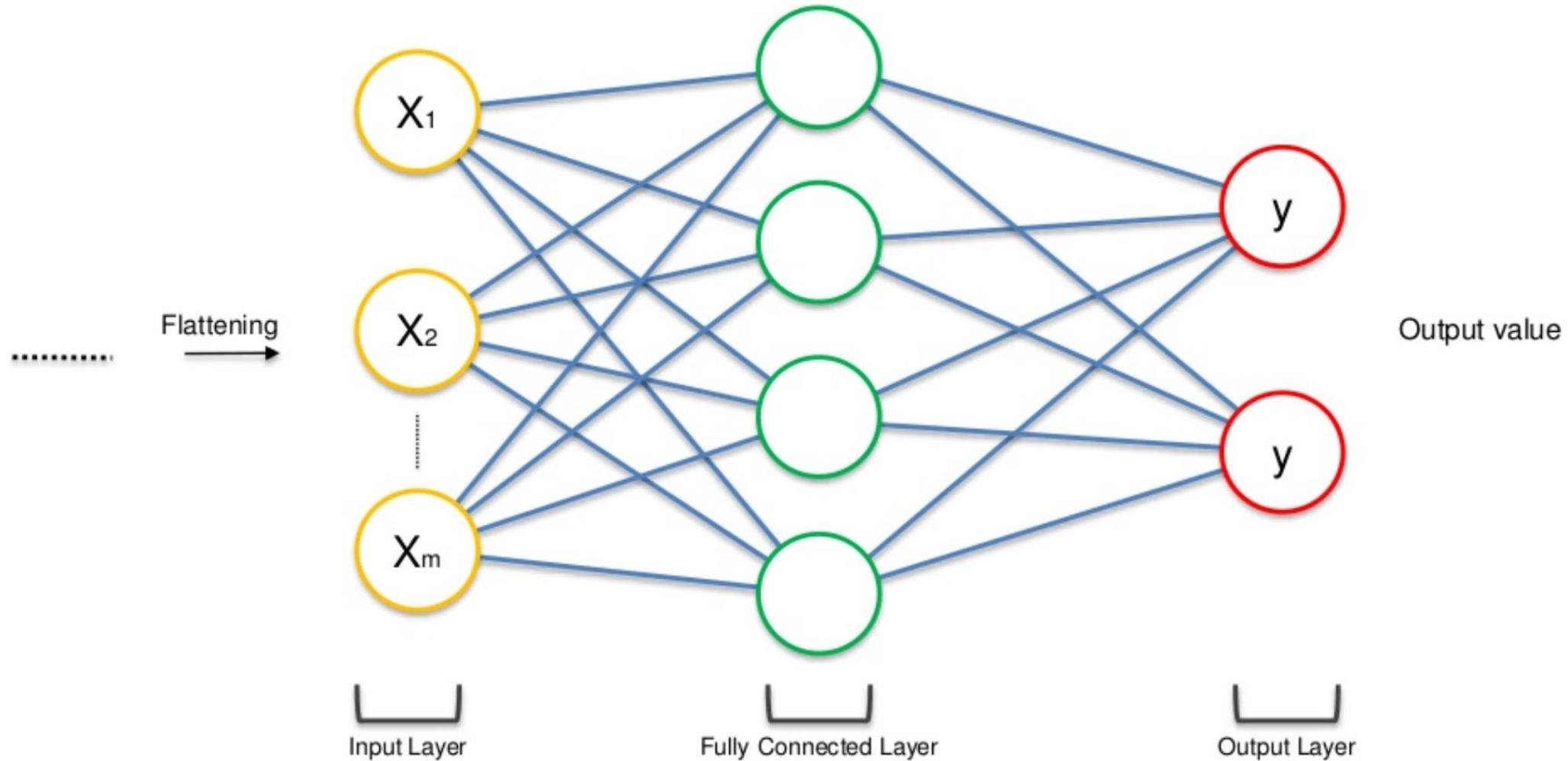
Flattening

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

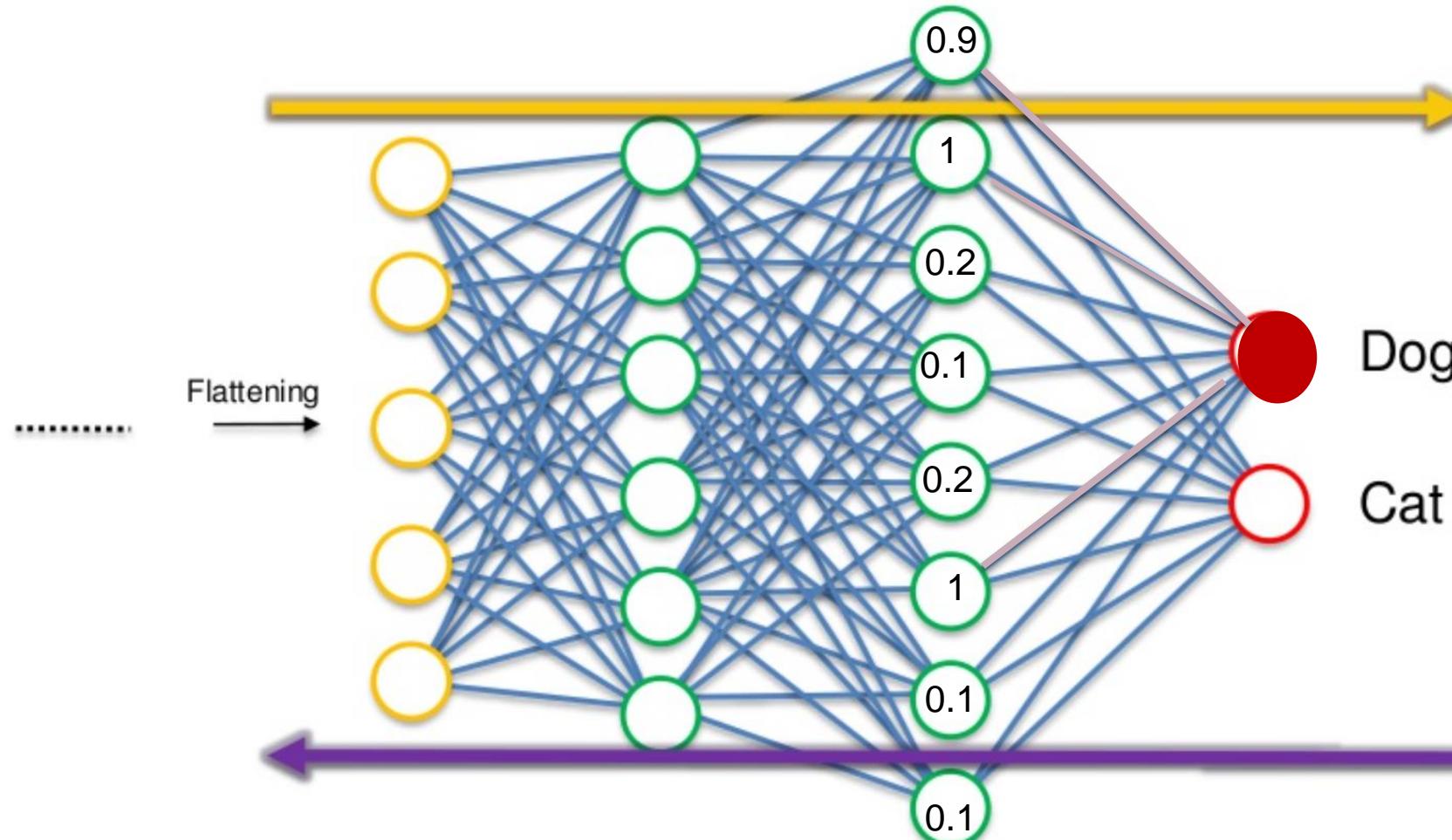
Input Image



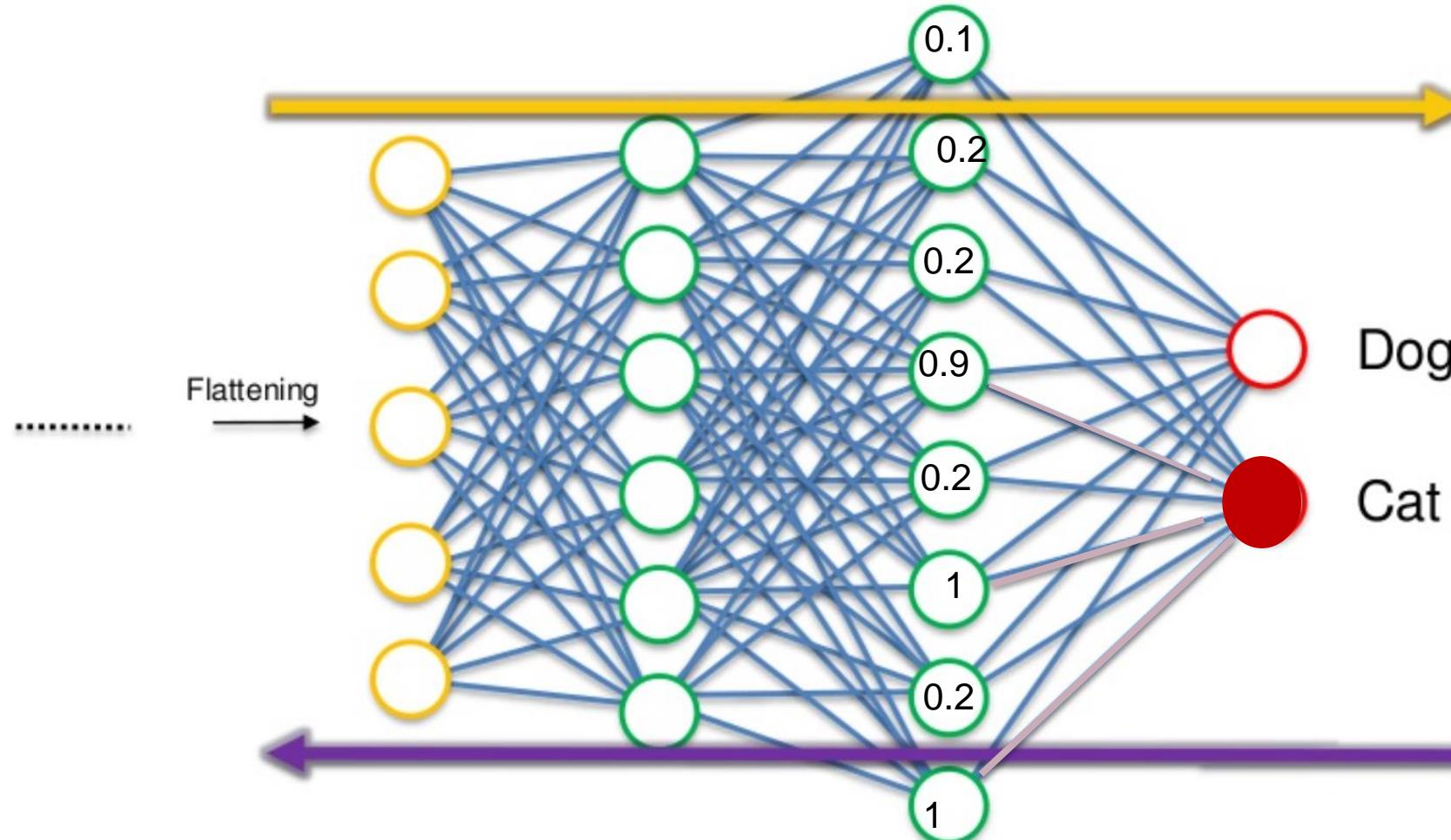
Full Connection



Full connection



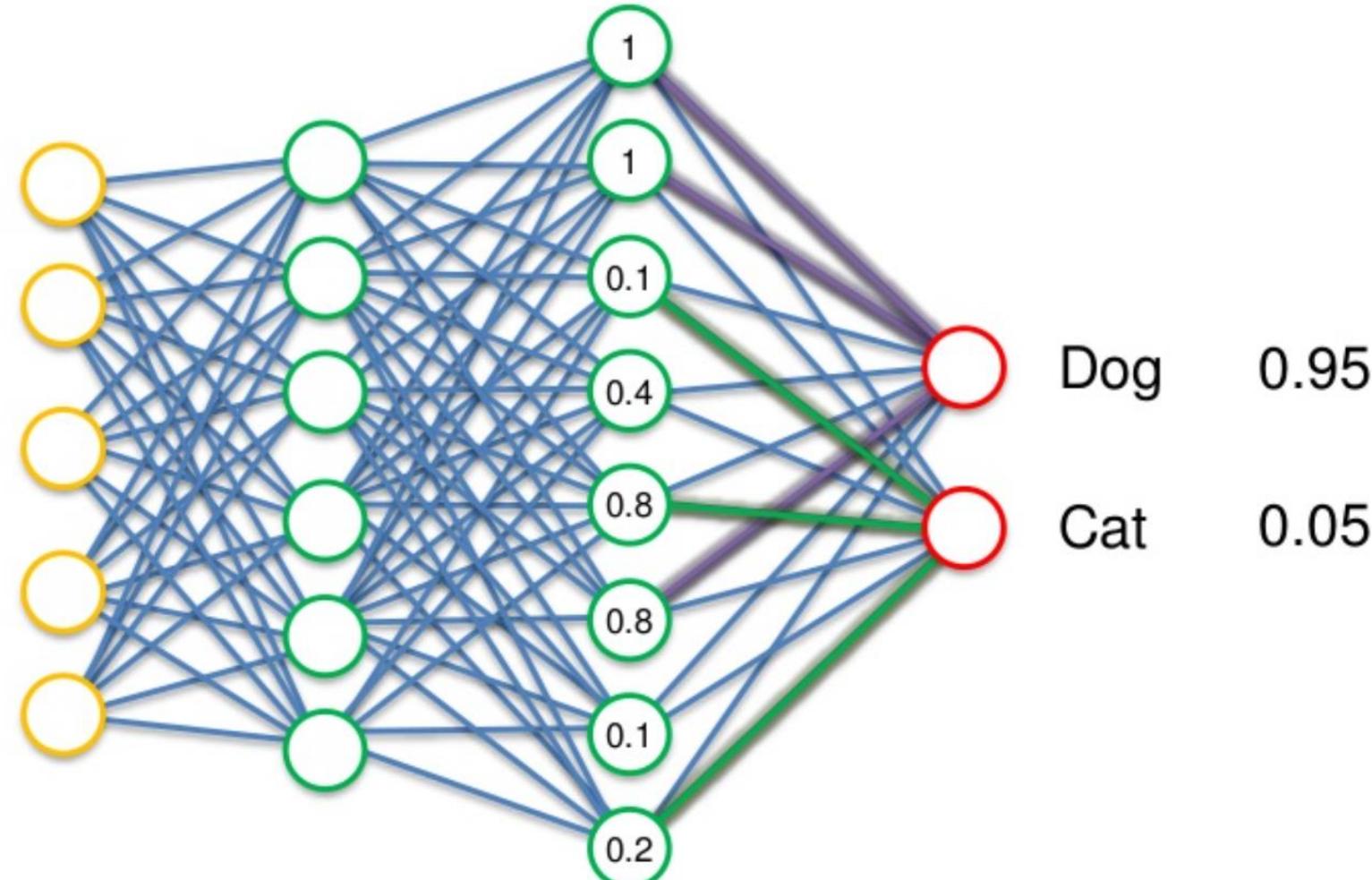
Full connection



Full connection



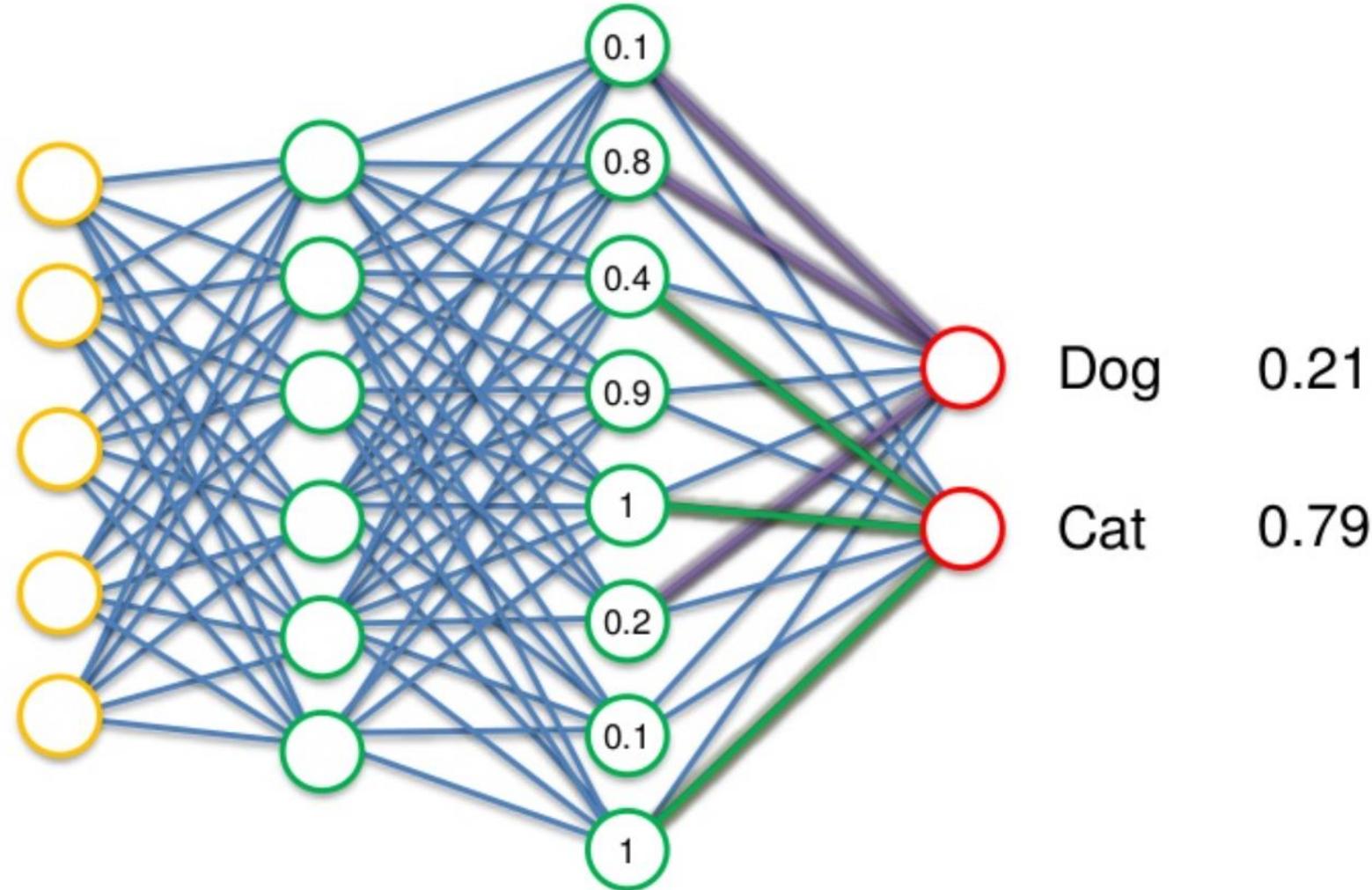
..... $\xrightarrow{\text{Flattening}}$



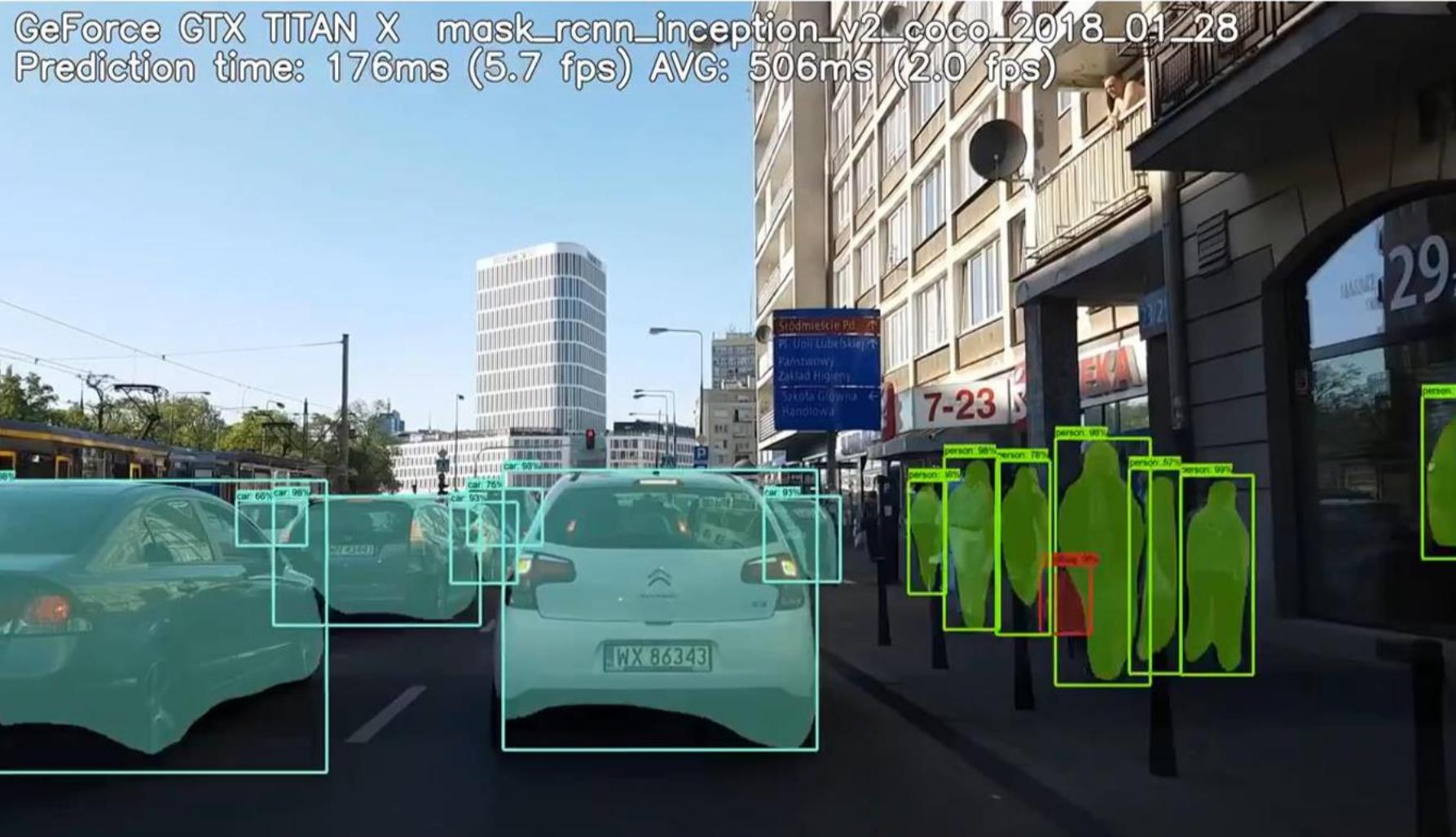
Full connection



.....
Flattening \longrightarrow

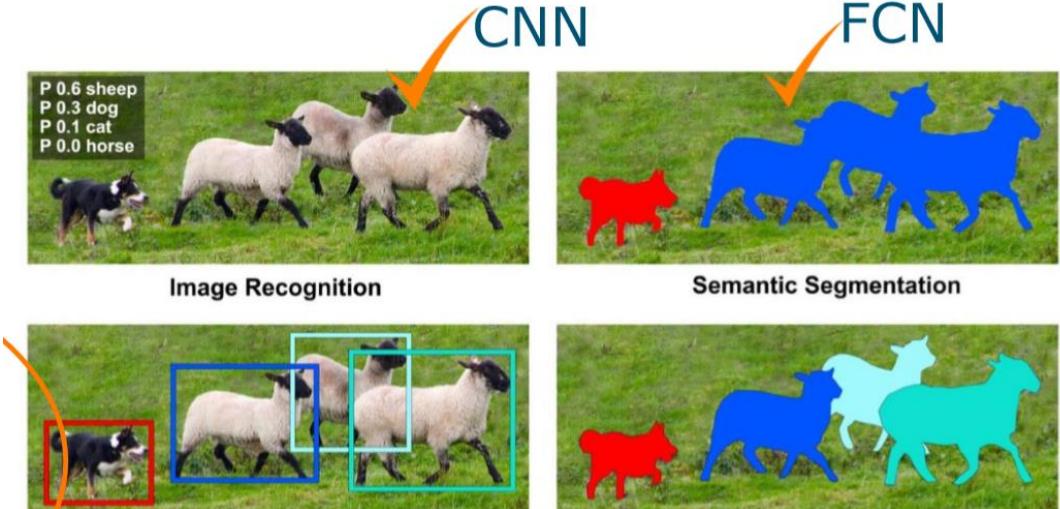


Deep learning: Object detection and segmentation

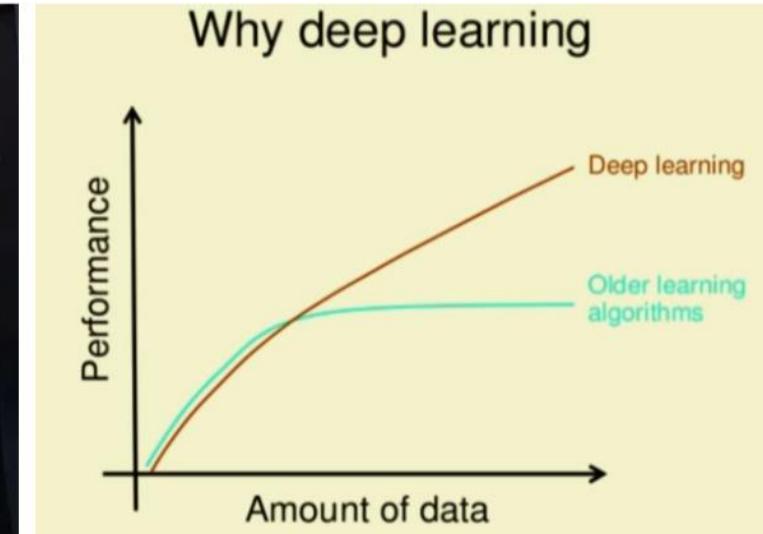
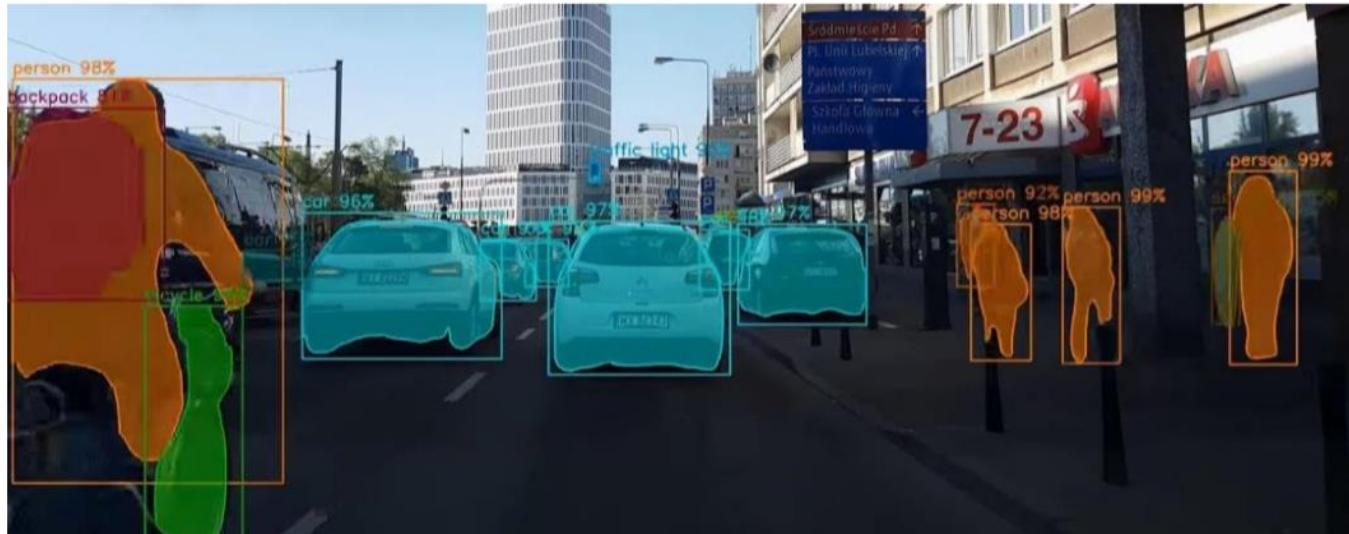


Object Detection and Instance Segmentation

- What is 'object detection and instance segmentation'?
- **Classification:** Is there a sheep or a dog in the image?
- **Semantic Segmentation:** Which pixels are sheep and dog pixels?
- **Object Detection:** Where are the individual sheep and dogs in this image? **These are extremely hard computer vision problems**
- **Instance Segmentation:** Where are the sheep and dogs and which pixels belong to them



Why is deep learning successful now?



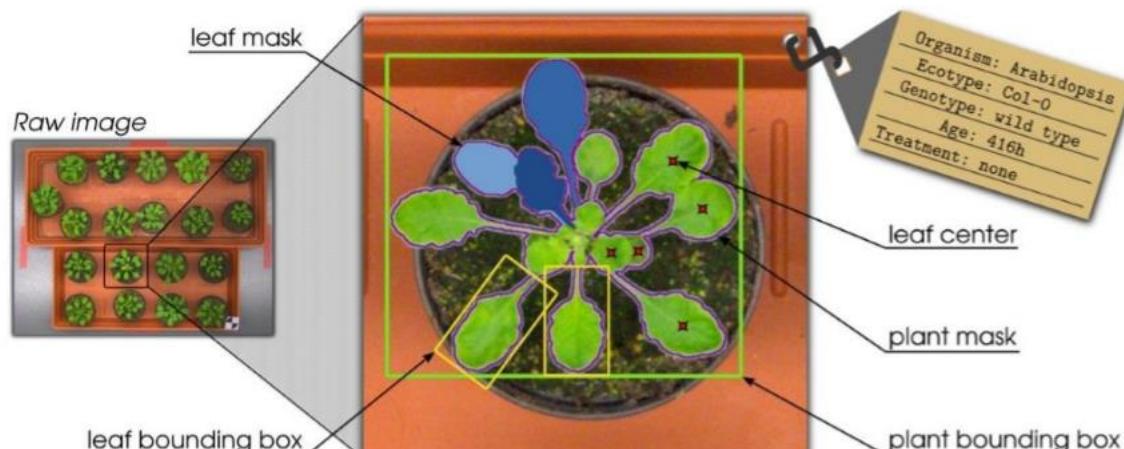
- In the last decade, deep learning has greatly outperformed classical machine learning in a multitude of very challenging problems.
- Availability of more data

Why is deep learning successful now?

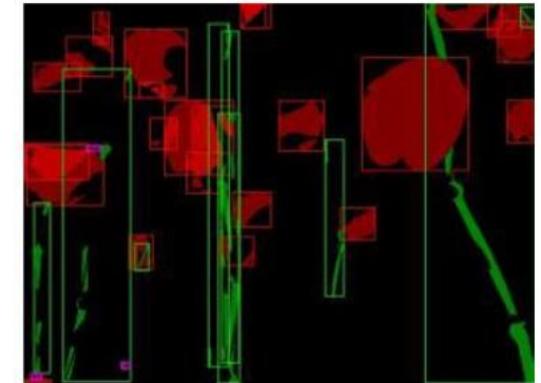
- ❖ Better network design choices (for example, convolutional nets)
- ❖ Better approaches to training a network
- ❖ Massive amounts of data (for certain tasks)
- ❖ Computing hardware much more powerful and better suited to task (GPUs)

Deep learning: For plant phenotyping

- Common phenotypical aspects of interest
- Structural: Height, width, volume ...
- Physiological: Carbohydrate content, water content, stress level ...
- Temporal: Leaf and stem elongation time, germination time, fruit maturity ...



Detecting and segmenting leaves in *Arabidopsis*



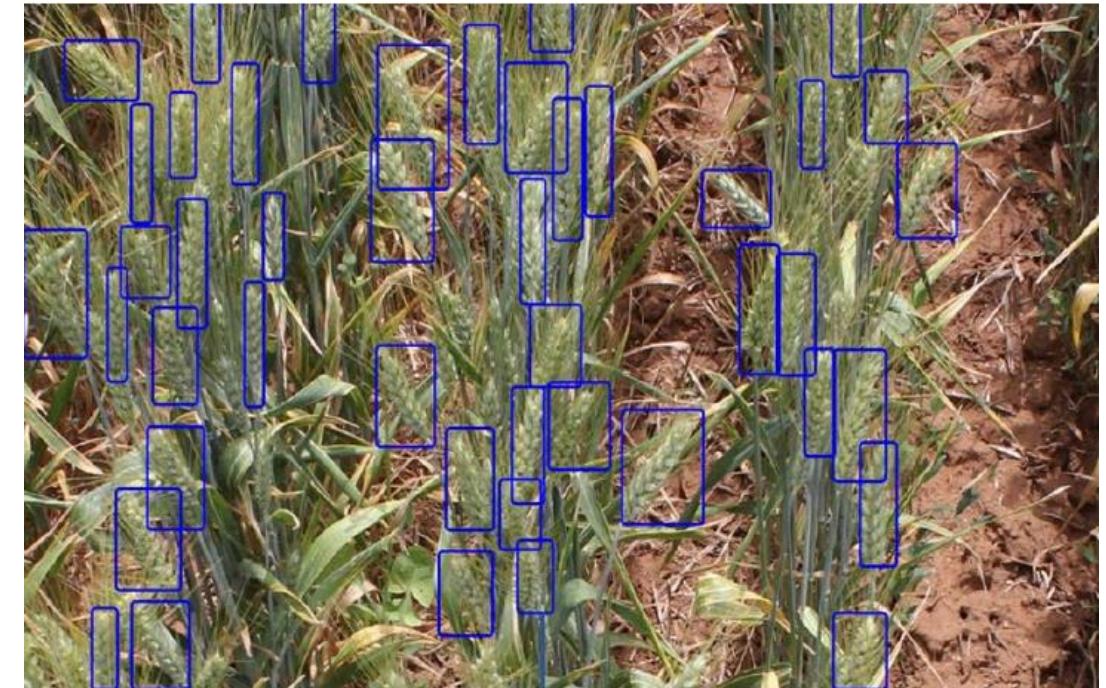
Detecting and segmenting peppers, leaves and stems in greenhouses

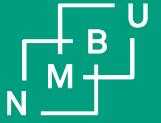


Detection and segmentation of tulip breaking virus

Deep learning: For plant phenotyping

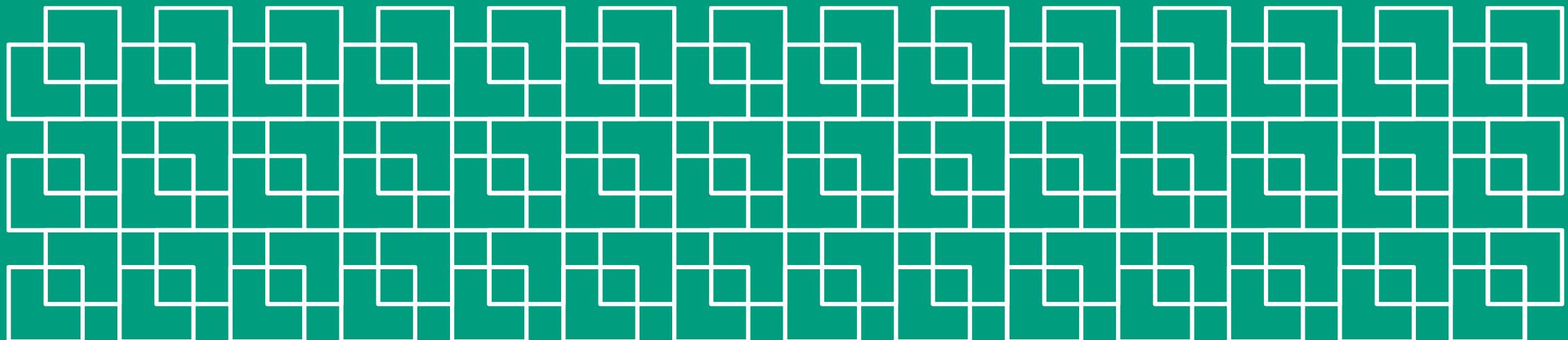
- Counting the number of spikes and estimation of grain yield
- [promo.mp4](#)





Question

sahameh.shafiee@nmbu.no



Quiz INF250

1. How many intensities are there in a 8-bit image?

| | |
|----|--|
| A) | 2 |
| B) | 16 |
| C) | 255 |
| D) | 256 v |

2. Which of these operations will enhance the edges in the image?

| | |
|----|--|
| A) | Thresholding |
| B) | Laplace filter. v |
| C) | Median filter |
| D) | Histogram equalisation |

3. Which of these is not morphological filter in image analysis?

| | |
|----|---|
| A) | Dilation |
| B) | Opening |
| C) | Infiltration v |
| D) | Closing |

4. What is a Sobel filter?

| | |
|----|---|
| A) | A smoothing filter |
| B) | A point operation |
| C) | An edge detecting filter v |
| D) | A filter computing the area of an object |

5. What is the effect of histogram equalisation?

| | |
|----|---|
| A) | Increasing the brightness |
| B) | Increasing the contrast. v |
| C) | Removing noise |
| D) | Increasing the lightness |

6. What is an erosion of an image?

| | |
|----|---|
| A) | Computes the local minimum over the area of a given kernel v |
| B) | Gives the area of an object |
| C) | Computes the local maximum over the area of a given kernel |
| D) | Filters the noise |

7. Which of these are edge detecting filters?

| | |
|----|--|
| A) | Canny v |
| B) | Median |
| C) | Sobel v |
| D) | Gaussian |

8. Which of these are not point operations?

| | |
|----|--|
| A) | Gaussian filter v |
| B) | Histogram equalization |
| C) | Canny filter v |
| D) | Median filter v |

9. What is an unsharpen mask?

| | |
|----|---|
| A) | A smoothed image divided by the original image |
| B) | A smoothed image multiplied with the original image |
| C) | The difference between a smoothed image and the original image v |
| D) | A thresholded image minus the original image |

10. What is represented by the 0th order moment of an object ?

| | |
|----|---|
| A) | The area of the object v |
| B) | The length of the major axis of the object |
| C) | The perimeter of the object |
| D) | The circularity of the object |