Basics and Hands-on of Computer Vision

Intensive course at University of Helsinki – day 1

Jorma Laaksonen

Aalto University
Department of Computer Science
Espoo

14.5.2018

Contents

Contents

Practical arrangements

- Approx 45 minutes lecture every morning
- Approx 15 minutes break after the lecture
- The hands-on work will be in Google Drive after the lecture
  - `https://tinyurl.com/UH-CV-18`
- Lecturer will be present to approx noon for questions
- Home work will be released when first ones are ready with hands-on or around 11 o'clock
- Also hands-on work can be done at home
- Results of both are reported in a single PDF that is due midnight
- Report also how long time the assignments took

Contents

Introduction

- The slides are based on two books:
  - Gonzalez & Woods: Digital Image Processing
    `http://imageprocessingplace.com/`
  - Sonca, Hlavac & Boyle: Computer Vision
- Short history of OpenCV
  - 1999 Intel Research initiative
  - 2000 first alpha version in ICCV conference
  - 2006 version 1.0
  - 2009 version 2.0 with C++ interface
  - 2015 version 3.0
  - native Python interfaces

3. Introduction

3.1 What computer vision stands for? (1)

- qualitative / quantitative explanation of images
- structural / statistical recognition of objects

3.2 What for is computer vision needed?

- quality control in manufacturing
- medical diagnostics
- robot control
- surveillance cameras
- analysis of remote sensing (satellite) imagery
- intelligence/espionage applications
- image databases
- optical character recognition
- biometrics

3.3 Why is computer vision difficult? (1.2/)

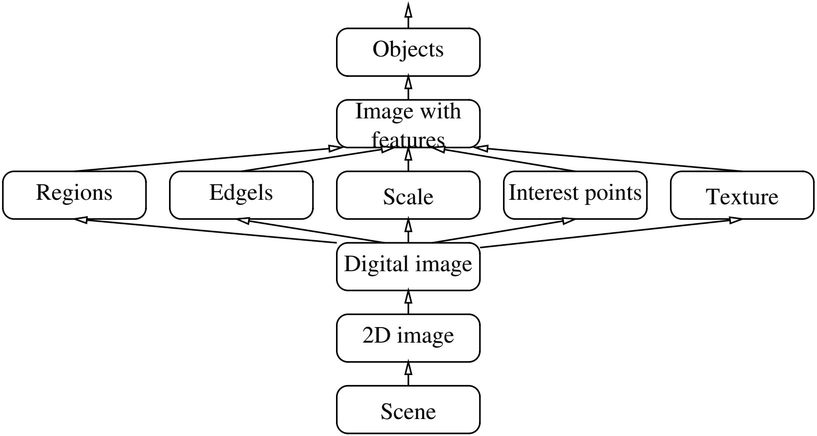
- loss of information in 3D $\rightarrow$ 2D projection
- interpretation of data by a model is problematic
- noise is inherently present in measurements
- there is way too much data
- measured brightness is weakly related to world's properties
- most methods rely on local analysis of a global view

3.4 What are the essential parts of a CV system?

- low-level image processing
  - noise reduction
  - sharpening
  - edge detection
  - scale, rotation and location normalization
  - compression
  - feature extraction
- segmentation
- high-level "understanding"
  - model fitting
  - hypothesis testing
  - classification
  - feedback to preprocessing

3.5 Image representation and analysis (1.3/)

Many different intermediate image content representations can be used.



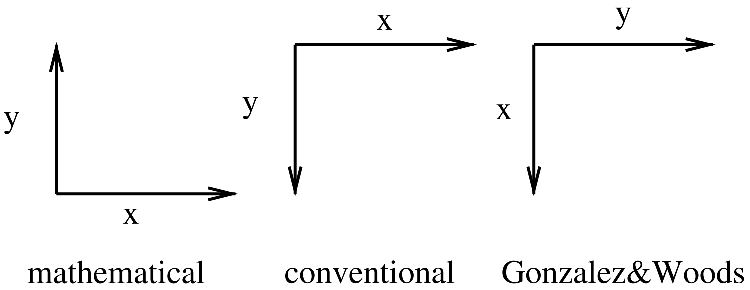
Contents

2.3 Representing digital images (2.4)

Coordinate systems (2.4.2)

A digital image is typically presented as a function of spatial x and y coordinates.

The orientation of the coordinate axes vary from book to book.



Sampling and quantization (2.4.2)

Digitizing of spatial xy coordinates represents two-dimensional sampling, also known as spatial quantization.

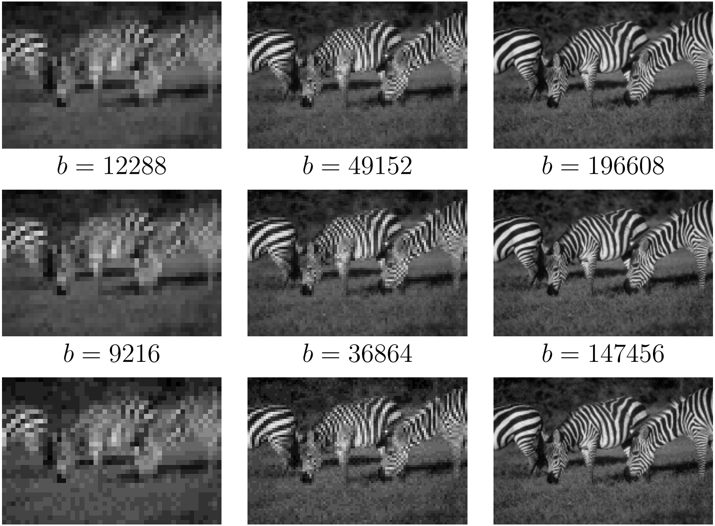
Digitizing of intensity amplitude is known as gray scale or intensity quantization.

A digital image is presented as an $M \times N$ matrix:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

One needs to set the spatial resolution $M \times N$ and the gray scale resolution L. The resolutions are often powers of two: $M = 2^m$, $N = 2^n$, $L = 2^k$. Storing of an image will then require b bits where: $b = M \times N \times k$.

The quality of television image can be obtained when $M = N = 512$ and $k = 7$.



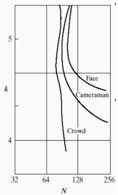
	48×32	96×64	192×128
256	$b = 12288$	$b = 49152$	$b = 196608$
64	$b = 9216$	$b = 36864$	$b = 147456$
16	$b = 6144$	$b = 24576$	$b = 98304$

Subjective image quality (2.4.3)

Image resolutions and bit counts do not match directly with the human subjective impression of the image quality.

Subjective assessments can be studied using **isopreference curves**.



Images that contain large even areas (low frequencies) require many intensity quantization levels to avoid false contouring.

Images that contain many small details (high frequencies) require good spatial resolution.

4. Digital image

4.1 Basic properties and definitions (2.1)

- continuous / discrete / digital image
- intensity / depth image
- monochromatic / multispectral image
- photometry: intensity, brightness, gray levels
- colorimetry: analysis of color (wavelength) information
- resolution: spatial / spectral / radiometric / temporal

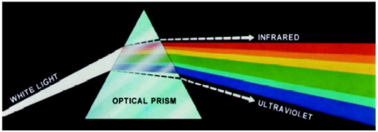
4.2 Digitization of images (2.2)

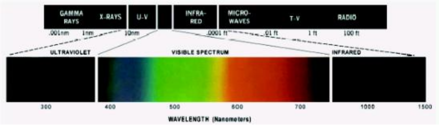
- sampling
- resolution
- 2D sampling interval $\Delta x, \Delta y$
- sampling points, sampling grid
- band-limited spectrum
- Shannon's sampling theorem
- quantization

Contents

14.2 Color fundamentals (6.1)

The **color spectrum** consists of **pure colors**, i.e. colors that consist of radiation of one wavelength only.





Six wide bands can be identified:

purple	380–450 nm
blue	450–480 nm
green	480–570 nm
yellow	570–600 nm
orange	600–670 nm
red	670–760 nm

Primary colors (6.1)

Wavelengths that correspond to **red** (R), **green** (G) and **blue** (B) are considered as the **primary colors**, because they match the human eye cones and they can generate the widest scale of combined colors. **All colors cannot be produced from the RGB primaries.**

The primary colors have been standardized: blue 435.8 nm, green 546.1 nm and red 700 nm. These don't fully match the human eye physiology.

Color television and LCD and LED monitors use three sub pixels whose colors match the above wavelengths and whose combined radiation creates the sensed color.

Secondary colors (6.1)

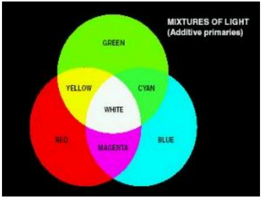
Secondary colors mean **pigment colors**: **cyan**, **magenta** and **yellow**.

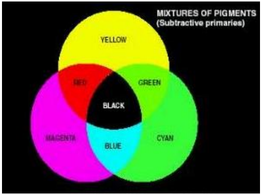
Each secondary color can be created as a sum two primary colors:

$$\begin{array}{ccccc} \text{red} & + & \text{blue} & = & \text{magenta} \\ \text{green} & + & \text{blue} & = & \text{cyan} \\ \text{red} & + & \text{green} & = & \text{yellow} \end{array}$$

The secondary colors are defined based on the wavelengths absorbed.

As a combination of all secondary pigment colors **black** is created. In printing the colors are generally created with three secondary colors and black.





Color sensing (6.1)

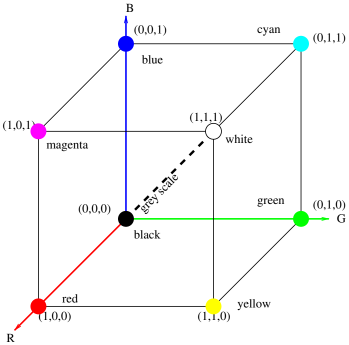
Sensing of colors is based on three physiological quantities:

- **brightness**
- **hue** characterizes the wavelength of the pure color matching the sensed color
- **saturation** characterizes the relative purity of the color, i.e. how much white/grey/black has been mixed with the pure color. The pure color of spectrum are fully saturated so they don't contain any mixed white.

The **chromaticity** of color means the combination of hue and saturation.

RGB color model (6.2.1)

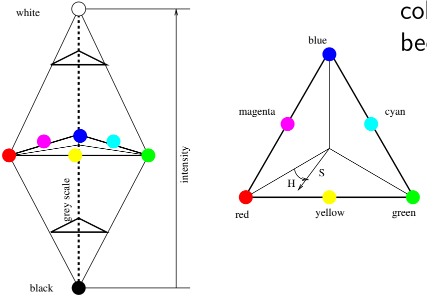
In the RGB color model red, green and blue are mutually orthogonal color axes and each component has values between zero and one. Black is in the origin and the cube diagonal originating from it forms the grey scale or intensity axis to white.



One can consider a color image to consist of three overlaid independent image planes or components.

RGB is a good choice for color presentation, but not so good choice for color image processing and color matching because it does not separate brightness from chromaticity.

HSI color model (6.2.3)



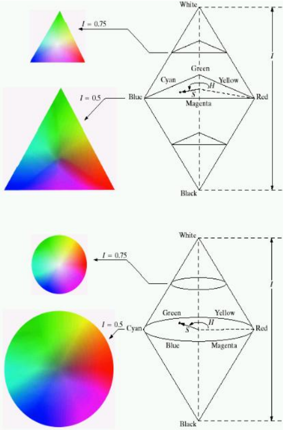
For most of color image processing the color model of choice is the HSI system because

- intensity I (over value V) is separated from chromaticity HS
- hue H is periodic, i.e. the ends of the spectrum (red and purple/-magenta) reside close to each other
- saturation S expresses the mixing of the pure color with white

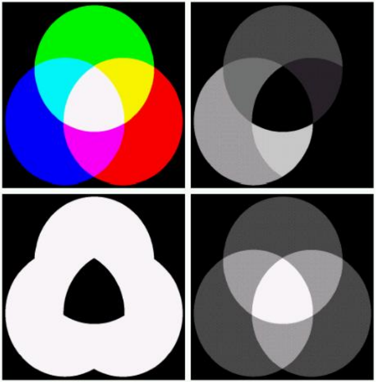
Transforms between RGB and HSI models are nonlinear and therefore somewhat difficult to implement. Also the periodicity of hue H can cause some problems.

HSI system axes and color circle (6.2.3)

The intersection of HSI axes with constant intensity can be drawn as a triangle, circle or hexagon.



hue



saturation intensity

CIE $L^*a^*b^*$color model (6.5.4)

The CIE $L^*a^*b^*$color model derived from the tristimulus values X, Y and Z is the best choice for precise color matching because

- it separates intensity from chromaticity,
- it is **colorimetric**, i.e. close-by colors have close-by values and color dissimilarity can be measured with Euclidean distance,
- it is **perceptually uniform**, i.e. the differences in hue values are perceived uniformly.

$$L^* = 116\, h\left(\frac{Y}{Y_W}\right) - 16 \qquad (X_W, Y_W, Z_W) = \text{reference white}$$

$$a^* = 500\left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right)\right]$$

$$b^* = 200\left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right)\right]$$

$$h(q) = \begin{cases} \sqrt[3]{q} & q > 0.008856 \\ 7.787q + 16/116 & q \le 0.008856 \end{cases}$$

14.5 Color transforms (6.5)

The general form of color transforms is the same as with spatial intensity images:

$$g(x,y) = T[f(x,y)]$$

Now $f(x,y)$ and $g(x,y)$ are vector valued and $T[\cdot]$ similar to that in intensity transform operations where the neighborhood contains only pixel itself.

$$s_i = T_i(r_1, r_2, \ldots, r_n), \qquad i = 1, 2, \ldots, n$$

s_i and r_i refer to the color components in the used color coordinate system, e.g. RGB, CMYK tai HSI. Remember that H is periodic in HSI!

An example: multiplication of image intensity with constant k:

$$g(x,y) = k\, f(x,y)$$

HSI	RGB	CMY
$s_1 = r_1$ $s_2 = r_2$ $s_3 = k\, r_3$	$s_i = k\, r_i \;,\; i = 1,2,3$	$s_i = k\, r_i + (1-k) \;,\; i = 1,2,3$

Color slicing (6.5.3)

In color slicing one creates a pseudocolor image that emphasizes some colors and/or dim others.

The preferred color is defined e.g. in the RGB space by its center (a_1, a_2, a_3) and the length l of the cube side or the radius R_0 of the sphere around it:

$$s_i = \begin{cases} 0.5 & \text{if } \sum_{j=1}^{n}(r_j - a_j)^2 > R_0^2 \\ r_i & \text{otherwise} \end{cases}$$

cube

sphere

Contents

4.3 Metric properties of a digital image (2.3.1)

- distance $D(p,q)$ is a metric iff:

  1) $D(p,q) = 0 \Leftrightarrow p = q$ (identity)

  2) $D(p,q) > 0 \Leftrightarrow p \neq q$ (non-negativity)

  3) $D(p,q) = D(q,p)$ (symmetry)

  4) $D(p,q) \leq D(p,r) + D(r,q) \;\; \forall r$ (triangular inequality)

- distances $D(p,q)$ between points $p = (i,j)$ and $q = (h,k)$:

$$
\begin{aligned}
D_E((i,j),(h,k)) &= \sqrt{(i-h)^2 + (j-k)^2} \\
D_4((i,j),(h,k)) &= |i-h| + |j-k| \\
D_8((i,j),(h,k)) &= \max\{|i-h|, |j-k|\} \\
D_{QE}((i,j),(h,k)) &= \max\{|i-h|, |j-k|\} + \\
&\qquad (\sqrt{2}-1)\min\{|i-h|, |j-k|\}
\end{aligned}
$$

Distance transform aka chamfering algorithm

1) pixel p in the object: $F(p) := 0$, otherwise $F(p) := \infty$

2) scanning top to bottom, left to right, causal 4- or 8-neighborhood AL:

$$F(p) := \min_{q \in AL}(F(p), D(p,q) + F(q))$$

3) scanning bottom to top, right to left, causal 4- or 8-neighborhood BR:

$$F(p) := \min_{q \in BR}(F(p), D(p,q) + F(q))$$

AL	AL	BR
AL	p	BR
AL	BR	BR

AL and BR

1	1	1
1	p	1
1	1	1

$D_8()$

2	1	2
1	p	1
2	1	2

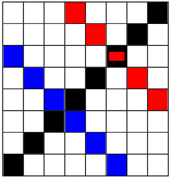
$D_4()$

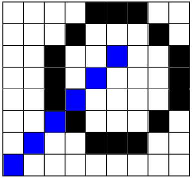
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	1	1	0	0	0	1	0
0	1	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0

5	4	4	3	2	1	0	1
4	3	3	2	1	0	1	2
3	2	2	2	1	0	1	2
2	1	1	2	1	0	1	2
1	0	0	1	2	1	0	1
1	0	1	2	3	2	1	0
1	0	1	2	3	3	2	1
1	0	1	2	3	4	3	2

Adjacency of pixels

- 4- or 8-neighbors of pixels
- segmentation into regions on basis of adjacency
- path between pixels: simple/non-simple/closed
- contiguous pixels have a path between them
- being contiguous: reflective, symmetric and transitive
- simple contiguous = no holes, multiple contiguous = has holes
- connectivity paradoxes



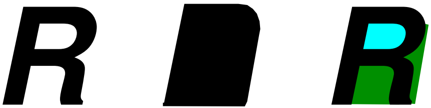


Segmentation, borders/boundaries and edges

- segmentation: region / object / backround / holes
- border/boundary is related to binary images
- edges are local properties of grayscale images: strength and direction
- crack edge: interpixel difference between 4-neighbor pixels

Topological properties

- rubber sheet and rubber band operations and invariances
- convex hull and its deficits: lakes and bays



Histograms (2.3.2)

Contents

4. Intensity transformations for enhancement

4.1 Basic notation for spatial operations (3.1.1)

The operations on pixels in the *image plane* or *spatial domain* can be denoted:

$g(x, y) = T[f(x, y)]$
$f(x, y)$ is the original input image
$g(x, y)$ is the resulting output image
$T[\cdot]$ is the operation applied to f in the neighborhood of point (x, y)

The operator T can be applied also to a set of aligned images or channels of a single image. In that case one could replace the scalar $f(x, y)$ with a vectorial $\mathbf{f}(x, y)$.

If T is applied to the (x, y) pixel alone, it is a *point operation*, otherwise it is a *mask operation*.

4.3 Histogram processing (3.3)

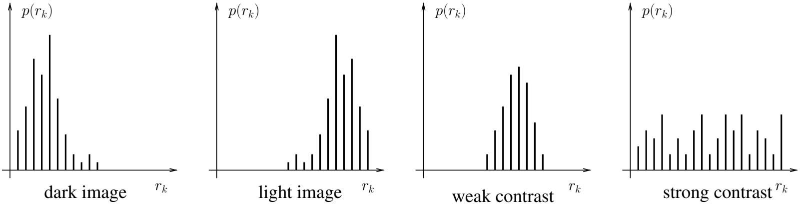
Histogram operations are an important class of intensity transformations.

The histogram of an image is formed by counting how many times each intensity value appears in the image:

$$
\begin{aligned}
p(r_k) = & \text{ is estimated probability of } r_k = \frac{n_k}{MN} \\
r_k \in [0, L-1] & \text{ is } k\text{:th discrete intensity value} \\
n_k & \text{ is the count of } k\text{:th intensity values in the image} \\
MN & \text{ is the total count of pixels in the image}
\end{aligned}
$$

Based on the shape of the histogram, the image's appearance can be described and the needed enhancement operations can be planned.

Examples of the forms of intensity value histograms (3.3)



It is often the most convenient to think that the intensity value r gets real values from range $[0, 1]$, where 0 corresponds to black and 1 to white.

Intensity histogram transformations (3.3)

Mappings that can be used for transforming intensity histograms are mostly of the form $s = T(r)$, where $T(r)$ is

- unique and monotonically increasing in the range $0 \leq r \leq 1$, which ensures that the ordering of intensity values does not change
- $0 \leq T(r) \leq 1$, when $0 \leq r \leq 1$, which ensures that the resulting intensity values remain within bounds

The inverse transformation $r = T^{-1}(s)$ exhibits the same good properties.

In the continuous case one can study the differentials:

$$p_s(s) = \left[p_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)}$$

This shows that the transformed image's intensity histogram $p_s(s)$ can be forced to any shape by a proper selection of $T(r)$.

Intensity histogram equalization (3.3.1)

The most commonly used histogram operation aims at making all intensities equally probable. This can be obtained by selecting the transfer mapping:

$$s = T(r) = \int_0^r p_r(w)\,dw\ , \quad 0 \le r \le 1$$

The right side of the equation is r's cumulative distribution function (CDF). CDF is monotonically increasing from 0 to 1.

We can easily solve the derivative of s with respect to r:

$$\frac{ds}{dr} = p_r(r)$$

Its inverse $\frac{dr}{ds}$ can be inserted in the previous equation:

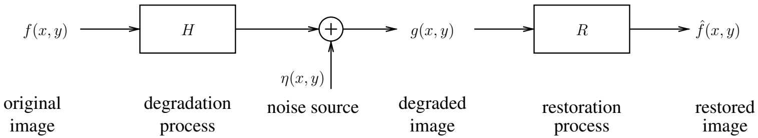
$$\begin{aligned} p_s(s) &= \left[p_r(r)\frac{dr}{ds}\right]_{r=T^{-1}(s)} = \left[p_r(r)\frac{1}{p_r(r)}\right]_{r=T^{-1}(s)} \\ &= 1\ , \quad 0 \le s \le 1 \end{aligned}$$

This shows that $s = T(r)$ produces an equalized histogram $p_s(s)$.

Contents

9.1 Degradation model (5.1)

Let us assume the following (*degradation/restoration model*):



The process H is assumed to be linear and position invariant, noise η is assumed to be uncorrelated and additive. Therefore the process H can be interpreted as a convolution and the convolution theorem leads to:

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

We will first assume that $H = 1$ and study only the effects of additive noise. Later we will study the degradation model in full.

5. Spatial filtering for image enhancement

5.1 Spatial mask operations (3.4/3.5)

Many digital image processing techniques are based on performing arithmetic (or logic) operations in a small fixed neighborhood of each pixel.

The operations are called as mask operations, template operations, window operations, filtering operations, convolution operations, ...

Arithmetic neighborhood operations can be expressed by using the intensity values z_i of the image and the mask coefficients w_i.

$$\begin{array}{c|c|c|c|c}
 & & \vdots & & \\
\hline
 & z_1 & z_2 & z_3 & \\
\hline
\cdots & z_4 & z_5 & z_6 & \cdots \\
\hline
 & z_7 & z_8 & z_9 & \\
\hline
 & & \vdots & &
\end{array}
\qquad
\begin{array}{|c|c|c|}
\hline
w_1 & w_2 & w_3 \\
\hline
w_4 & w_5 & w_6 \\
\hline
w_7 & w_8 & w_9 \\
\hline
\end{array}$$

As an example, a 3×3-sized mask $w(x, y)$ for calculating the average:

$$z = \frac{1}{9}(z_1 + z_2 + \cdots + z_9) = \frac{1}{9}\sum_{i=1}^{9} z_i$$

Linear operations and operators (2.6.2/2.6)

A central concept in image processing is whether an operation is linear or not.

Let's assume that an image operation H transforms an input image f to output image g:

$$H[f(x,y)] = g(x,y)$$

Operator H is said to be linear, iff

$$H[a_i f_i(x,y) + a_j f_j(x,y)] = a_i H[f_i(x,y)] + a_j H[f_j(x,y)]$$

Linear operator thus has two properties: *additivity* and *homogeneity*.

Operation that is not linear is by definition *nonlinear*.

Linear filtering as spatial convolution (3.4.2/4.2.4)

Linear filtering operations can be interpreted as convolutions.

Definition of a convolution:

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

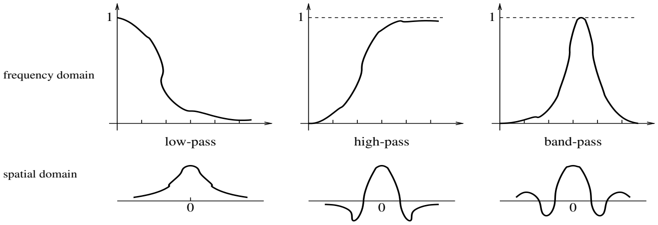
Convolution is *commutative*:

$$w(x,y) \star f(x,y) = f(x,y) \star w(x,y)$$

Shapes of linear spatial filters

The *impulse response* or *point spread function* of a linear filter shows the spreading of a single white pixel on black background. The shape of the impulse response equals the point reflection (central inversion) of the mask's weights.

The *Fourier transform* of the impulse response is known as the *filter transfer function* (or simply *filter function*). Linear filters are commonly circular symmetric in both spatial and frequency domains. 1D intersection of the impulse response gives information on the filter's frequency domain properties.



Examples of linear smoothing (3.5.1/3.6.1)

(500x500)

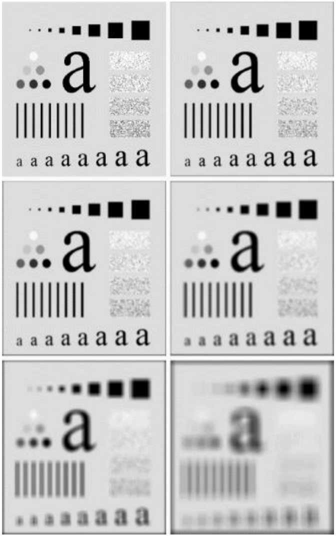
3x3

5x5

9x9

15x15

35x35



Order-statistics filters (3.5.2/3.6.2)

If the new intensity value for the output image is obtained as something else than a linear combination of the intensity values in the neighborhood of the pixel in the input image, then the operation is nonlinear.

Lowpass filtering can be implemented with nonlinear order-statistics filters. Order-statistics filters are based on the *ranking* of the intensity values under the mask. The operations include:

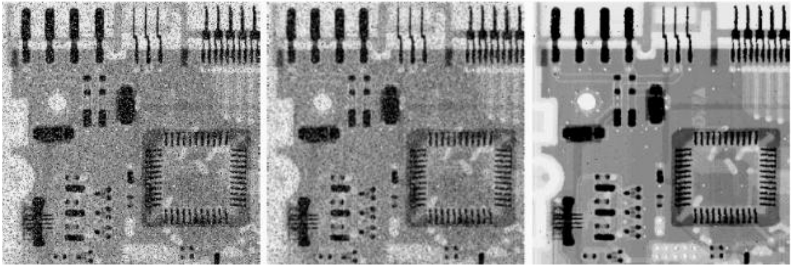
- median
- maximum
- minimum

Nonlinear filters like the median filter do not have a known impulse response nor a transfer function. One may say that median filtering is unique for each image.

Noise removal with median filtering (3.5.2/3.6.2)

Linear neighbor averaging as a noise removal method tends to blur details. This can to some extent be prevented by using median filtering. Also median filters destroy image details, but not as much as linear lowpass filters of same size.

Median filtering is an optimal method for removing strong *impulse noise*, known also as *salt and pepper noise*.



9.3 Restoration in the spatial domain (5.3)

If the degradation is caused by only additive (non-periodic) noise, the restoration is simplest in the spatial domain. If the noise is periodic or the degradation model contains a true degradation process, then the restoration is easiest in the frequency domain.

Noise can be removed in the image domain by averaging. In addition to the arithmetic (ie. linear) average, there exist also other variants:

$$\text{arithmetic mean } \hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

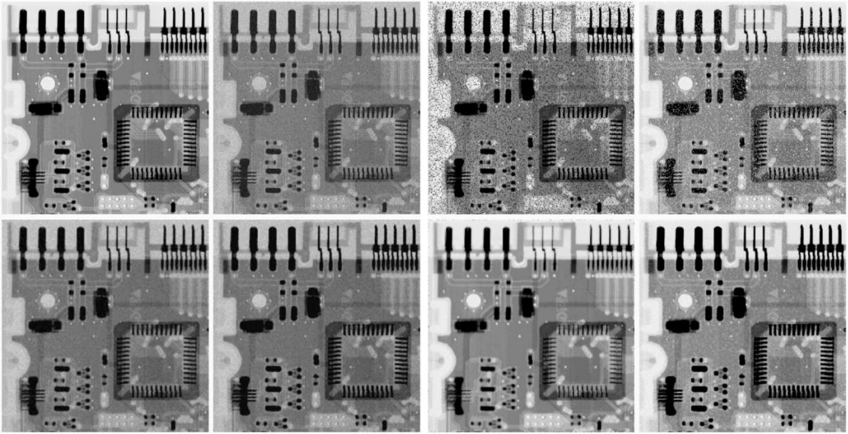
$$\text{geometric mean } \hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

$$\text{harmonic mean } \hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

$$\text{contraharmonic mean } \hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

Examples of averaging (5.3.1)

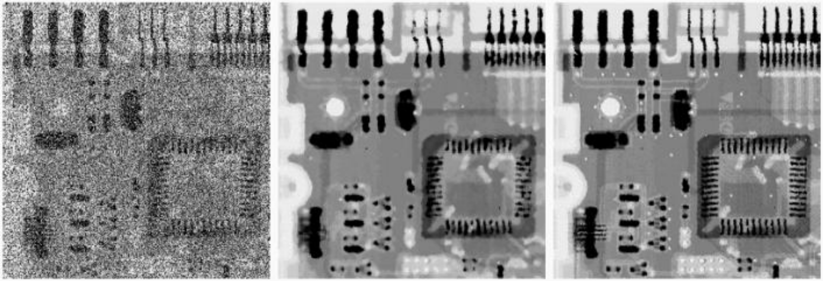
original Gaussian noise pepper noise salt noise



arithmetic mean geometric mean $Q = 1.5$ $Q = -1.5$

Adaptive median filtering (5.3.3)

One can modify the size of the median filtering mask on the basis of the properties of the image under the mask. One increases the mask size if $z_{\text{med}} = z_{\text{min}}$ or $z_{\text{med}} = z_{\text{max}}$, so that z_{med} is free from the impulse noise. If finally $z_{xy} = z_{\text{min}}$ or $z_{xy} = z_{\text{max}}$, then z_{med} is used as the output, otherwise z_{xy}.



impulse noise $\quad$ 7×7 median $\quad$ adaptive, $S_{\text{max}} = 7$.

5.3 Image sharpening with highpass filtering (3.6/3.7)

Image sharpening aims to enhance blurred details in images or to highlight transitions in intensity. Sharpening can be interpreted as inverse operation of averaging.

Sharpening is based on amplifying the intensity differences between pixels. Derivatives (or differences in the discrete case) are well suited for detecting interpixel changes.

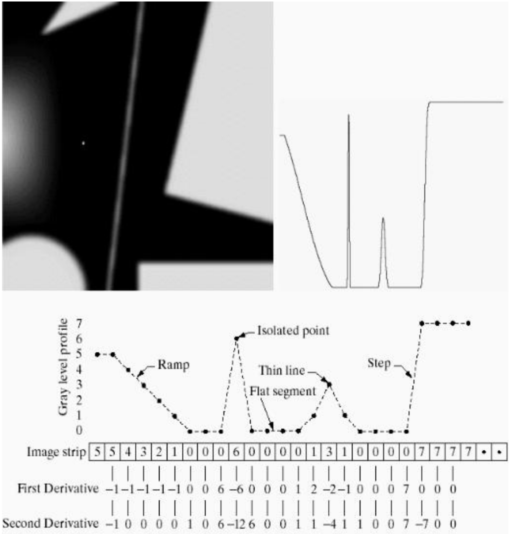
The first-order derivative (actually difference) of a one-dimensional function:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

The second-order derivative of a one-dimensional function:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Examples of types of image details (3.6.1/3.7.1)



The Laplacian (3.6.2/3.7.2)

For a continuous two-dimensional function, the Laplacian is defined as:

$$\Delta f(x, y) = \nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

We can see that the Laplacian is a linear operator.

In the one-dimensional case, the first derivative was defined as:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$
$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$
$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

In the mask form:

0	1	0
1	-4	1
0	1	0

or

1	1	1
1	-8	1
1	1	1

Sharpening with the Laplacian (3.6.2/3.7.2)

The Laplacian enhances small details and vanishes (equals zero) for constant and linearly varying areas. Laplacian filtered image can be added to the original one:

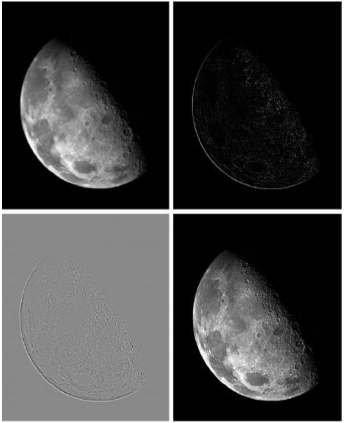
$$
\begin{aligned}
g(x,y) &= f(x,y) - \nabla^2 f(x,y) \\
&= 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)
\end{aligned}
$$

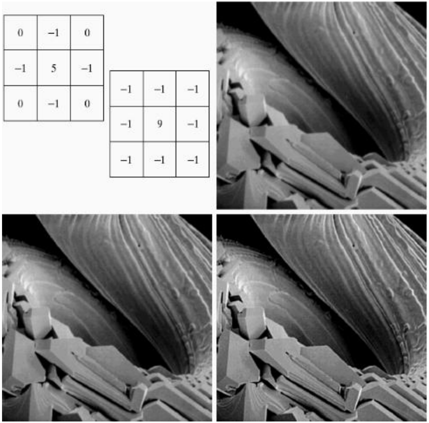
$$
\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}
-
\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}
=
\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}
$$

or:

$$
\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}
-
\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}
=
\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 9 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}
$$

Sharpening with the Laplacian, an example (3.6.2/3.7.2)





Contents

Hands-on work and home assignment

- Instructions in Google Drive
- Also hands-on work can be done at home
- Report all code and images of the hands-on and the home assignment in the same PDF
- Report also how long time the assignments took
- Email the PDF before midnight to the lecturer