# Grundlagen Bildverarbeitung
# Imaging & Image Processing

Univ.-Prof. Dr. Andreas Uhl

WS 2016/2017

**Abstract**

The basis of this material are course notes compiled by students which have been corrected and adapted by the lecturer. The students' help is gratefully acknowledged.

Some of the images are taken from websites without prior authorization. As the document is only used for non-commercial purposes (teaching students) we hope that nobody will be offended!

# Contents

# List of Figures

# 1 Introduction

## 1.1 Literature on Image Processing

There are many books about the subject so here are some examples ...

- Fundamentals of Digital Image Processing, Anil K. Jain
- Fundamentals of Electronic Image Processing, Arthur R. Weeks Jr.
- Digital Image Processing, Rafael C. Gonzalez, Richard E. Woods
- Image Processing, Analysis, and Machine Vision, Milan Sonka, Vaclav Hlavac, Roger Boyle

The most important Journals for publication on the subject are ...

- IEEE Transactions on Image Processing (TIP)
- IEEE Transactions on Circuits and Systems for Video Technology
- Computer Vision and Image Understanding
- SPIE Journal of Electronic Imaging
- Image and Vision Computing
- Signal Processing: Image Communication
- International Journal on Imaging and Graphics
- IEEE Transactions on Medical Imaging[1]

Three of the most important conferences on image processing and computer vision ...

- IEEE International Conference on Image Processing (ICIP)
- SPIE Symposium on Electronic Imaging
- IEEE Computer Vision and Pattern Recognition (CVPR)
- IEEE International / European / Asian Conference on Computer Vision (I/E/ACCV)
- IEEE International Conference on Accustic, Speech and Signal Processing (ICASSP)

[1]image processing $\neq$ imaging!

## 1.2 Overview and Related Terms

### 1.2.1 Digital Image Processing

- Vision allows humans to perceive and understand the world surrounding us.
- Computer vision aims to duplicate the effect of human vision by electronically perceiving and understanding an image.
- Giving computers the ability to see is not an easy task – we live in a three dimensional (3D) world, and when computers try to analyze objects in 3D space, available visual sensors (e.g. TV cameras) usually give two dimensional (2D) images, and this projection to a lower number of dimensions incurs an enormous loss of information.
- In order to simplify the task of computer vision understanding, two levels are usually distinguished:

  **low level image processing** Low level methods usually use very little knowledge about the (semantic) content of images. Classical image processing.

  **high level image processing (understanding)** High level processing is based on knowledge, goals, and plans of how to achieve those goals. Artificial intelligence (AI) methods are used in many cases. High level image processing can be identified with computer vision and tries to imitate human cognition and the ability to make decisions according to the information contained in the image.

This course deals almost exclusively with (low level) image processing.

## 1.3 Low level digital image processing tasks

Low level computer vision techniques overlap almost completely with digital image processing, which has been practiced for decades. The following sequence of processing steps is commonly recognized:

**Image Acquisition** An image is captured by a sensor (such as a TV camera) and digitized

**Preprocessing** computer suppresses noise (image pre-processing) and maybe enhances some object features which are relevant to understanding the image. Edge extraction is an example of processing carried out at this stage

**Image segmentation** computer tries to separate objects from the image background

**Object description and classification** (in a totally segmented image) may also be understood as part of low level image processing, however, it is often seen as part of high level processing.

# 2 Image Acquisition and Representation

## 2.1 Human Visual System & Optical Principles

We see light – but what is that ? These three are the same .......

- Light: pure energy
- Electromagnetic waves: energy-carrying waves emitted by vibrating electrons
- Photons: particles of light

The electromagnetic spectrum as visualised in Fig. 1 is classically partitioned into the following seven bands (where for each waveform, the speed is identical, i.e. the speed of light (300,000,000 meters per second):

- Radio waves: communication
- Mirowaves: used to cook
- Infrared: "Heat Waves"
- Visible light: what humans see
- Ultraviolet: Causes sunburn
- X-Rays: Penetrate tissue
- Gamma rays: Most energetic

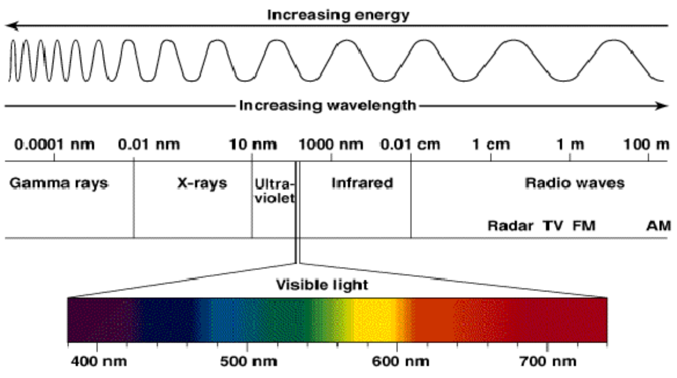


Figure 1: Electromagnetic spectrum

The human eye acts like a camera (or is a camera ?): The eye has an iris like a camera (used to control the aperture by radial muscels), focusing is done by changing the shape of the lens, but what is the sensor ? Photoreceptor cells (rods – "Stäbchen", and cones – "Zäpfchen") in the retina ! In Fig. 2 the location of the Fovea is shown, which is a small region of high resolution containing mostly cones. The optic nerve consists of 1 million flexible fibers used to transfer the acquired data for further processing. The region where the optic nerve leaves the retina is called the "disc', in this area there are no potoreceptors, thus, we are blind in this area !

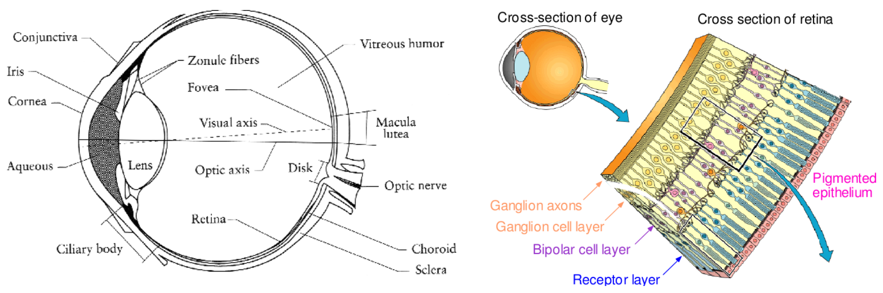


Figure 2: Human eye and retina.

Cones are less sensitive, operate in high light, and are respensible for colour vision. There are three types of them, which perceive different portions of the visible light spectrum: Red, green, and blue. Rods are highly sensitive, operate at night, and provide gray-scale vision only. It is interesting to note that these support peripheral vision only (see the distribution !) – so why are there more stars off-center ?

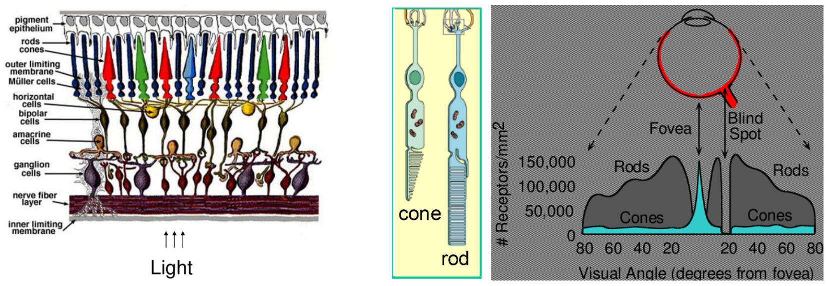


Figure 3: Color und luminance perception: Rods and cones.

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, the perceived intensity of a region is related to the light intensities of the regions surrounding it (see so called "Mach Bands" in Fig. 4).

Muscles within the eye are used to change the shape of the lens allowing us focus on objects that are near or far away. An image is focused onto the retina causing rods and cones to become excited which ultimately send signals to the brain. As shown in Fig. 5, the function of the lens has to be taken into account.

When trying to build a camera, we need some medium to emulate / replace the retina. In the last millennium, film was used which was able to record light due to a chemical reaction, nowadays, digital sensor took over. The first idea to build a camera is to put a sensor / film in front of an object like shown in Fig. 6.

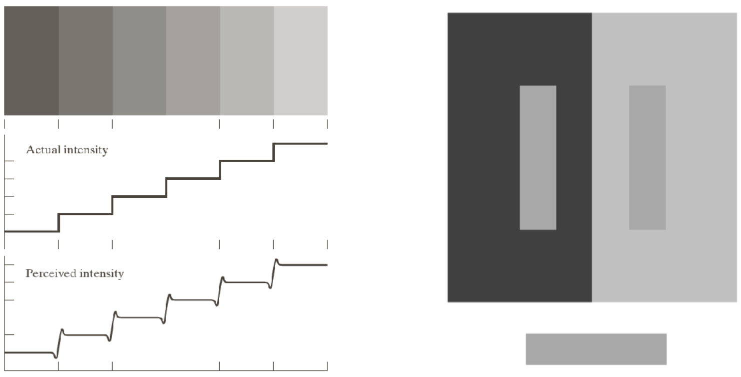


Figure 4: Perceived intensity depends on surroundings.

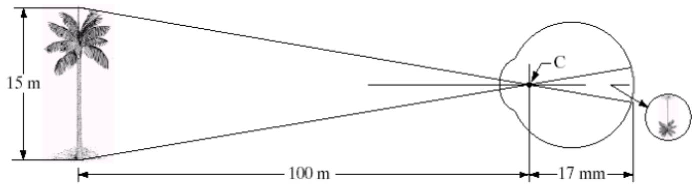


Figure 5: Image formation in the eye

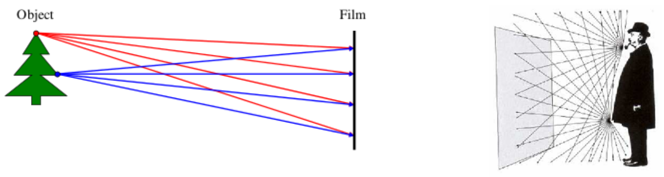


Figure 6: Using a plain sensor as a camera does not work.

We don't get a reasonable image as the information on the sensor is extremely blurred , basically each single object point is projected to each sensor element. This observation motivates the use of a barrier with a small aperture only, where a small amount of rays is able to pass. This concept is called “pinhole camera” which significantly reduces the blurring effect, the principle is visualised in Fig. 7.

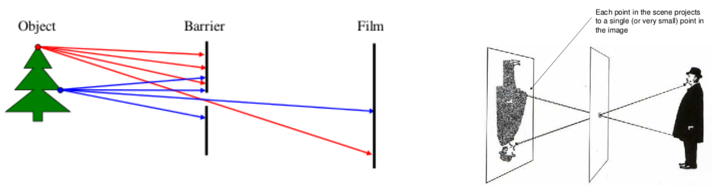

Object
Barrier
Film
Each point in the scene projects to a single (or very small) point in the image

Figure 7: Using a barrier with hole improves the situation.

Each point in the scene projects to a single point (or very small area) point the image. The focal length $f$ is the distance between the pinhole and the sensor as shown in Fig. 8. If we double $f$ we double the size of the projected object.

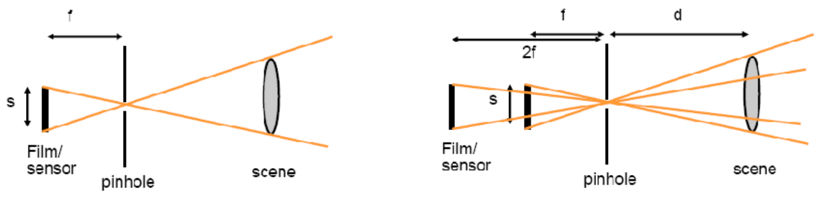
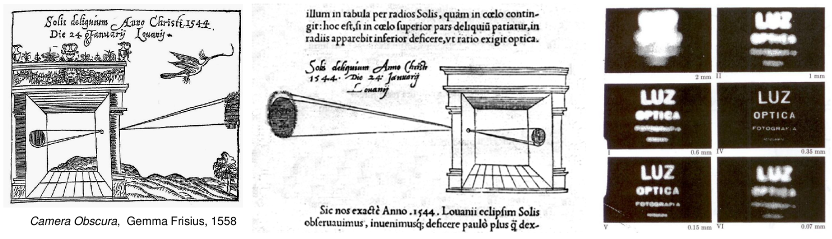


Figure 8: Influence of focal length on object size.

Although already known long ago (Aristoteles, "old" chinese history, etc.), pinhole cameras have severe limitations. The aperture (the "hole") must be very small to obtain a clear image. As a consequence, as the pinhole size is reduced, less light is received by the image plane (i.e. sensor). Further, if pinhole size is comparable to wavelength of incoming light, "diffraction" effects effects blur the image (as shown in Fig. 9). The strategy to obtain differently sized objects on the sensor is not very convenient (varying the barrier – sensor distance significantly).


Solis deliquium Anno Christi 1544.
Die 24 Januarij Louanij
Camera Obscura, Gemma Frisius, 1558
Solis deliquium Anno Christi 1544. Die 24 Januarij Louanij
LUZ
OPTICA
FOTOGRAFIA
2 mm
II
1 mm
I
0.6 mm
IV
0.35 mm
V
0.15 mm
VI
0.07 mm

illum in tabula per radios Solis, quàm in cœlo contingit: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in radiis apparebit inferior deficere, vt ratio exigit optica.

Sic nos exactè Anno .1544. Louanii eclipſim Solis obſeruauimus, inuenimusq; deficere paulò plus q̃ dex-

Figure 9: Historic pinhole cameras and tradeoff size of pinhole / sharpness.

For these reasons, the pinhole concept is replaced by introducing a lens (now we finally arrive very close to the human model), which focuses light onto the sensor. At a specific distance, objects are "in focus", other points project to a "circle of confusion" in the image, which results in blur. Changing the shape of the lens (as is done in the human eye) changes the focal distance. Parallel rays are focused onto a single point, the focal point. When lenses are used, $f$ denotes the distance between the plane of the lens and the focal point. An aperture of certain size restricts the range of rays passing through the lens and influences required exposure time (compare: pinhole size).

Changing the aperture also affects the depth of field: A smaller aperture increases the range in which the object is approximately in focus (we will see this in more detail, see Fig. 11). Usually, we consider lenses to be "thin" lenses, i.e. that the thickness of the lens is negligible compared to the focal length. In modern lenses, this is harldy the case !

In Fig. 12, the principle of generating clear and blurred images depending on the position of the sensor plane relative to the focal point and the lense plane is shown. We denote by $y$ the object size and by $y'$ the size of the object in the

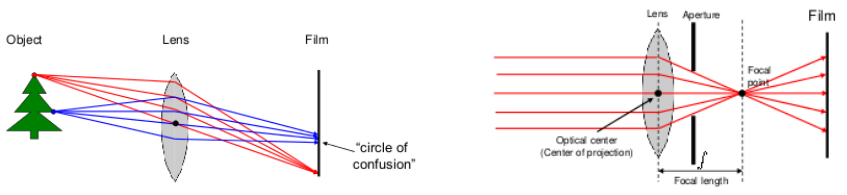


Figure 10: Introduction of the lens.

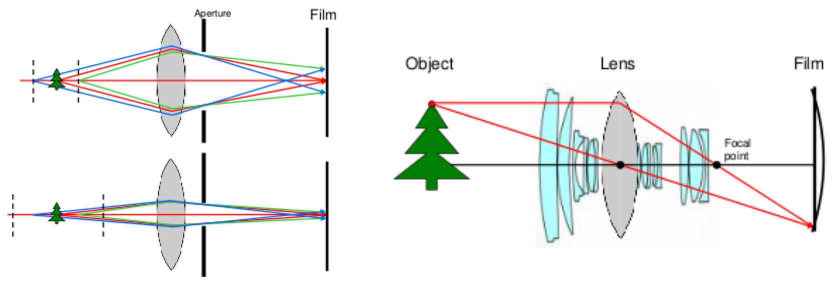


Figure 11: Effects of changing aperture and thick lenses.

image; further we denote by $d$ the distance between object and lens plane and by $d'$ the distance between sensor and lens plane. $f$ is the focal length.

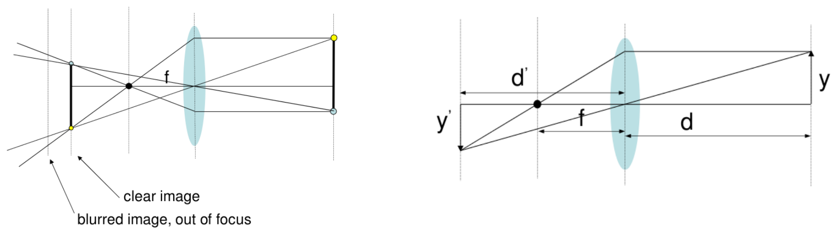


Figure 12: Thin lens scenario.

In Fig. 13, we visualise the principle of deriving the "thin lens formula" using simple geometric principles (i.e. triangle simularity properties). Exploiting the yellow triangles' similarity, we obtain

$$\frac{y'}{d'} = \frac{y}{d} \Longrightarrow \frac{y'}{y} = \frac{d'}{d} \; .$$

Similarly, exploiting the green triangles' similarity, we obtain

$$\frac{y'}{d'-f} = \frac{y}{f} \Longrightarrow \frac{y'}{y} = \frac{d'-f}{f} \; .$$

Equating the right side of both equations we result in

$$\frac{y'}{y} = \frac{d'}{d} = \frac{d'-f}{f} \Longrightarrow \frac{d'}{d} = \frac{d'}{f} - 1 \Longrightarrow \frac{1}{d} = \frac{1}{f} - \frac{1}{d'} \; .$$

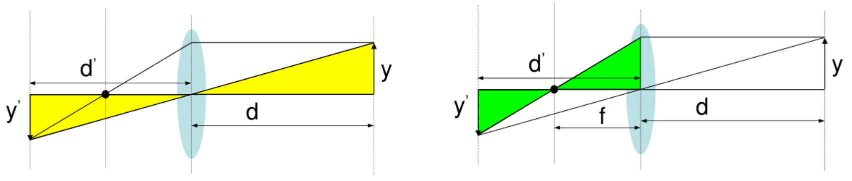


Figure 13: Deriving the thin lens formula.

This equation means that under thin lens assumptions objects are in focus for which the equation $\frac{1}{d'} + \frac{1}{d} = \frac{1}{f}$ is correct.

The consequences of this result for extreme situations are illustrated in Fig. 14: Objects at infinity focus at $f$: If $d \longrightarrow \infty$ then $d' \longrightarrow f$. By analogy, when the object gets closer, the focal plane moves away from $f$. At the limit: If $d \longrightarrow f$ then $d' \longrightarrow \infty$, i.e. an object at distance $f$ requires the focal plane to be at infinity.

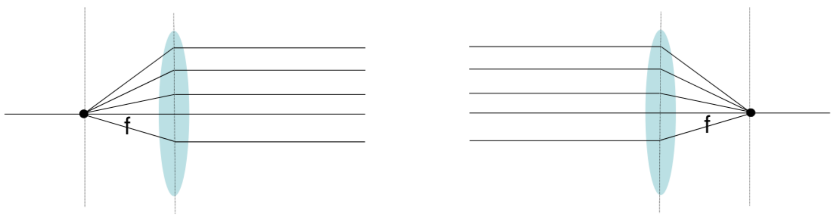


Figure 14: Extreme settings.

Varying the focal length $f$ (i.e. applying a zoom or changing the lens) results in a different image size as a consequence. We set $M$ as the relation between $y$ and $y'$, i.e. $M = \frac{y'}{y} = \frac{d'}{d}$. As a consequence of the thin lens equation ($\frac{1}{d'} + \frac{1}{d} = \frac{1}{f}$), $M = \frac{f}{d-f}$ for $d > f$, i.e. $M$ gets larger for increasing $f$ (see Fig. 15).

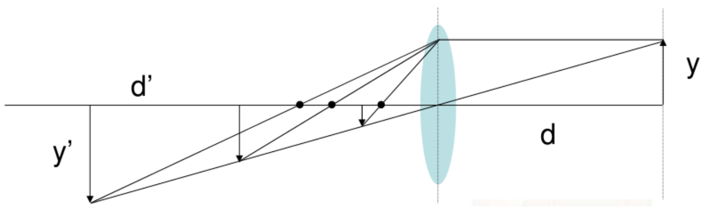


Figure 15: Effect of focal length on image size.

However, the sensor has a fixed size of course. As shown in Fig. 16, as $f$ gets larger, smaller parts of the world project onto the sensor, the image becomes more *telescopic*. On the contrary, as $f$ gets smaller, more world points project onto the sensor, the images become more wide angle in nature.

The depth of field ("Tiefenschärfe") is the range of object distances $d$ over which the object in the image are sufficiently well focused. In Fig. 17 the connection between the circle of confusion (the area of the sensor in the focal plane, onto which out of focus objects are projected) and the depth of field is visualised.

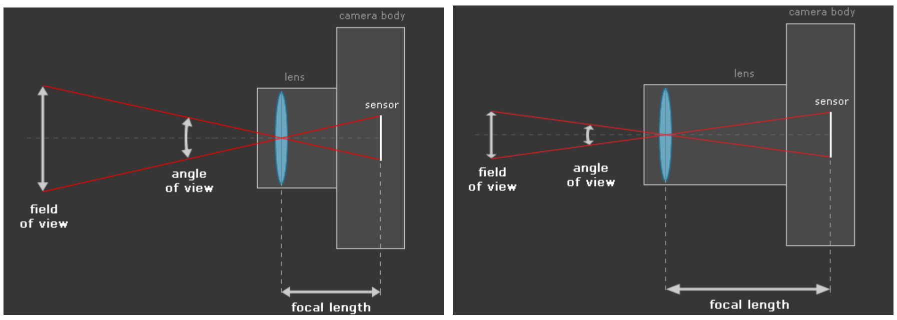


Figure 16: Effect of focal length on image content.

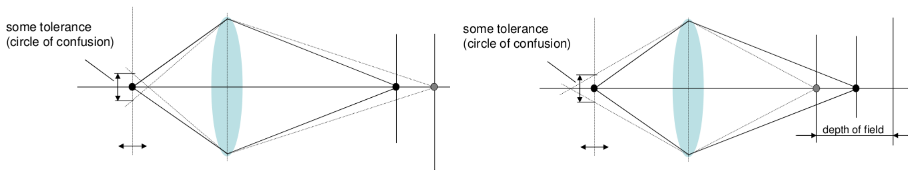

some tolerance (circle of confusion)
some tolerance (circle of confusion)
depth of field

Figure 17: Depth of field and circle of confusion.

Introducing the aperture changes the situation. Fig. 18 shows that when using an aperture, less rays arrive at the focal plane, so there is not much difference except that the image is darker (thus requiring a longer exposure time).

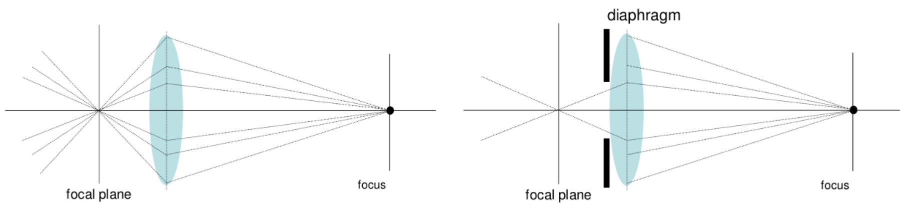


Figure 18: Effect of using an aperture (object in focus).

When considering the situation of out-of-focus points, the usage of the aperture has an additional effect: The circle of confusion is smaller, thus leading to a sharper (less blurry) image (see Fig. 19).

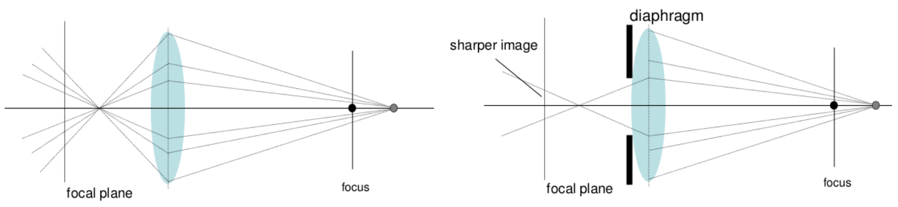


Figure 19: Effect of using an aperture (object non fucused).

In fact, it is possible to derive an explicit relation between the diameter of the circle of confusion $b$ (denoted as “blur circle” in Fig. 20) and the diameter of the aperture $d$ (note that object and image distance to the lens are inconsistently denoted as $o$ and $i'$ in the figure, respectively).

Based on observing similar triangles, $b = \frac{d}{i'}(i' - i)$ for two object distances $o, o'$ and their respective images distances $i', i$. Since the distance $(i' - i)$ cannot be determined reasonably, we want to express $b$ with $f$ and $o, o'$ instead. For both pairs $o, i'$ and $o', i$ the thin lens equation holds:

$$\frac{1}{i'} + \frac{1}{o} = \frac{1}{f} \text{ and } \frac{1}{i} + \frac{1}{o'} = \frac{1}{f} .$$

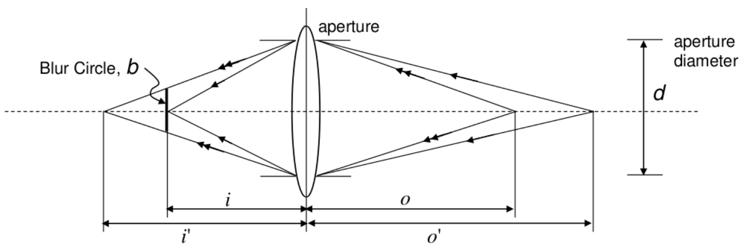


Figure 20: Computing the size of the circle of confusion.

From these formulas we can isolate $i$ and $i'$ and may derive

$$(i' - i) = \frac{f}{(o' - f)} \frac{f}{(o - f)} (o - o') .$$

Using this formula, we are able to compute the diameter of the circle of confusion based on objects' distance and the focal length $f$. This is important to exactly determine the exact depth of field – since this is the range for which the diameter of the circle of confusion is smaller than the resolution of the sensor. I.e. as long as all information within the circle of confusion is mapped to a single pixel, there is no blur.

Unfortunatley, several problems arise with the usage of lenses (see Fig. 21), some of which can be corrected, others are hard to correct (spherical aberration: Spherical lenses are the only easy shape to manufacture, but are not correct for perfect focus, different refractive indices leading to different focal points for different parts of the lens), and some of which may be used to identify certain specific lenses as done in image forensics.

## 2.2 Sensors

Besides classical stationary array sensors as found in digital cameras, also sensing devices with moving sensor elements and toroidal sensor areas have been developed as seen in Fig. 22. Moving sensor elements are found in scanners, while for “sweeping” fingerprint readers the finger is moved across a line of senor pixels.

In medical imaging a source of radiation is often moved to generate different perspectives of the data, in the shown tomography example to produce a set of

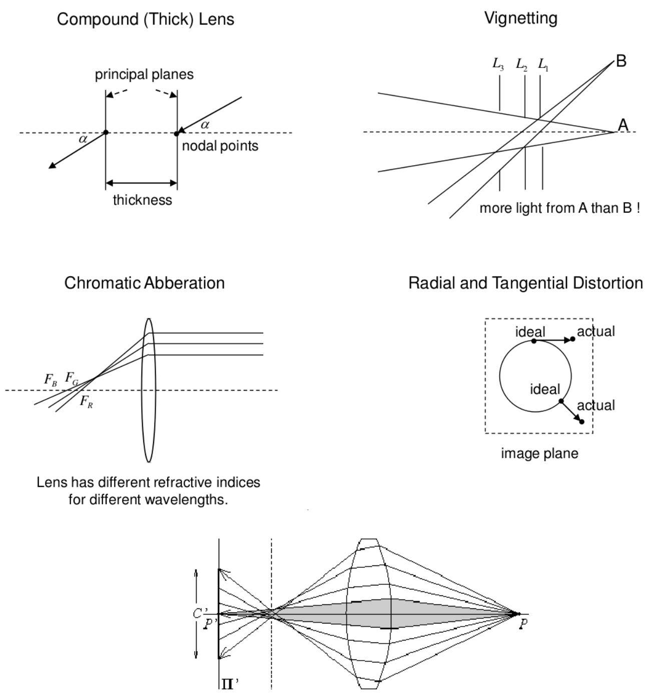


Figure 21: Some selected problems with lenses.

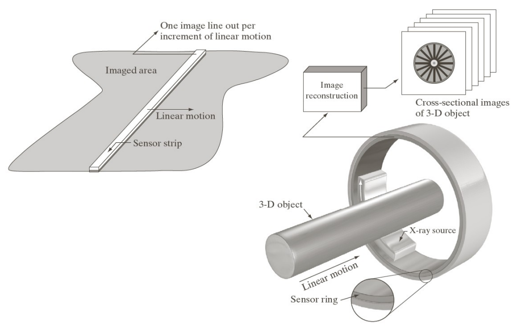


Figure 22: 1D and 3D sensing devices.

cross section images (but this can be applied to any organic material, e.g. also for generating cross section images of wood-logs).

The effect of radiation on the semiconductor is to kick electrons from the valence to the conducting band (still inside the material, see Fig. 23) – this effect can be read out and digitised. Classically, these electrons are collected and accumulated in the photosensitive cells, and once a control switch is activated, transfered out of these cells to be converted into measurable electrical quantities, i.e. charge/voltage. We require linearity between incoming light and resulting charge/voltage which is usually given, if not, a linearisation is being applied. Sensors with sensitivity against different bands of the electromagnetic spectrum can be constructed by using different types of semiconductor material. Silicium is well suited for the visible range and near-infrared (this is why we need to have near-infrared filters in our digital cameras), while material like e.g. Germanium and others are suited for the far-infrared range.

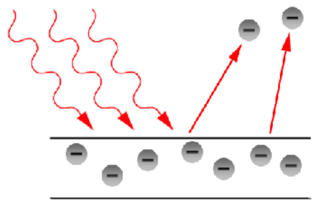


Figure 23: Photoelectric effect.

The two major types of sensors are CCD and CMOS sensors.

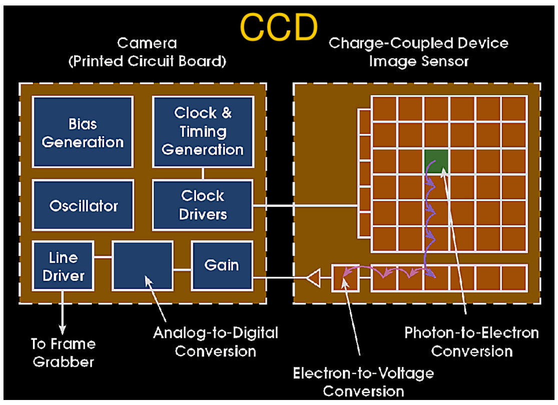


Figure 24: CCD sensor.

CCD sensor (for charge coupled device, see Fig. 24) consists in photosensitive cells able to store charge produced by the light-to-electron conversion; in addition, the charge can be transferred to an interconnected, adjacent cell. In this case charges are shifted out of the sensor (bigger sensors, better quality, but additional circuitry).

CMOS sensors (for complimentary metal oxide semiconductor, see Fig. 25), consist of transistors within the photossensitive cell which perform charge-to-voltage conversion and allow the pixels to be read individually (higher integration, less power consumption, but less sensitive, higher noise).

Fig. 26 shows a schematic comparison of the two technologies. The resulting different properties are summarised in Fig. 27.

There are different types of CCD configuartions / techniques which have some distinct properties which make them useful for different types of cameras / applications: full frame (FF), frame transfer (FT), interline transfer (IT) and frame interline transfer (FIT) (see also Figs. 28 and 30). These are frontside illuminated, that is, built so that the light enters on the same surface of the device that holds the circuitry, and they are almost all single-tap, having only

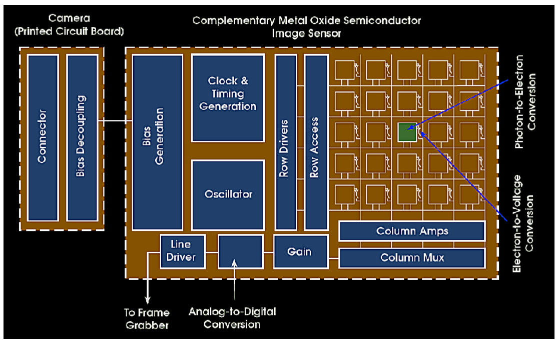


Figure 25: CMOS sensor.

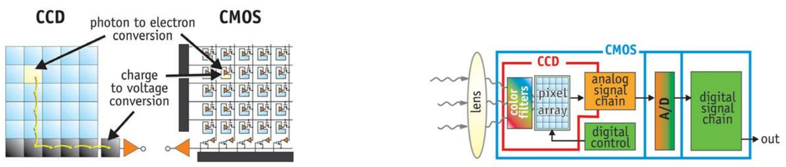
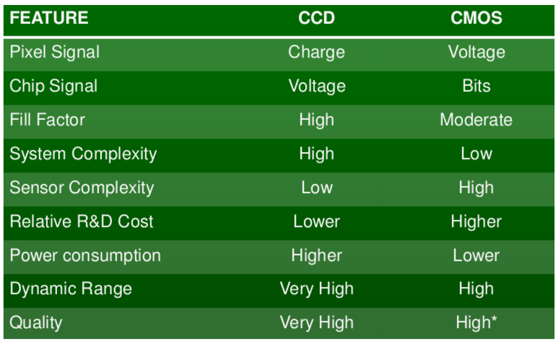


Figure 26: Schematic comparison of CCD and CMOS sensors.



| FEATURE | CCD | CMOS |
|---|---|---|
| Pixel Signal | Charge | Voltage |
| Chip Signal | Voltage | Bits |
| Fill Factor | High | Moderate |
| System Complexity | High | Low |
| Sensor Complexity | Low | High |
| Relative R&D Cost | Lower | Higher |
| Power consumption | Higher | Lower |
| Dynamic Range | Very High | High |
| Quality | Very High | High* |

Figure 27: Comparing CCD and CMOS sensors: Respective properties

one output point. In all of the figures, the light gray areas are sensitive to light and the dark gray areas are covered with aluminum. The arrows show the direction of charge motion when the electrodes are operating.

The Full Frame CCD image sensor (FF, see Fig. 28 left) is the simplest type. It consists of a set of photosensitive registers arranged next to each other in columns with a transport register at the bottom configured so that each well can receive charge from a different column. In a typical operating cycle, light is prevented from reaching the sensor by a shutter and then the charge is cleared from the photosensitive registers while the image sensor is in the dark. The shutter is opened for the desired exposure interval and then closed. In the dark, the charge in all of the columns is shifted down by one row so that the last row moves into the horizontal transport register. A faster clock then moves the charge packets in the horizontal transport register to the sampling node to generate the output voltage. When the row is complete, the next row is moved down for readout. This process is repeated until all rows have been read. The sensor is then ready for another exposure. Without shutter, a disadvantage is "charge smearing" (see Fig. 29) caused by light falling on the sensor whilst the accumulated charge signal is being transferred to the readout register. However, mechanical shutters have lifetime issues and are relatively slow. FF are typically the most sensitive CCD's available but charge readout is rather slow. There are two strategies to cope with the disadvantages of the FF CCD.

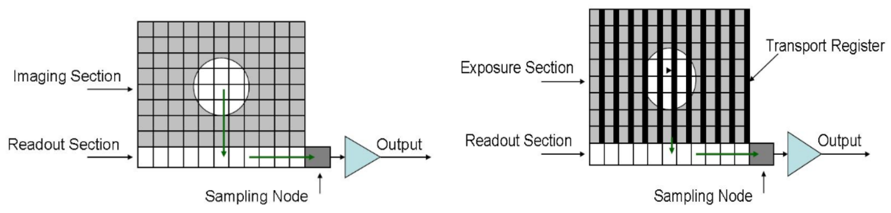


Figure 28: FF and IT CCD configurations.

The Frame Transfer architecture (FT, see Fig. 30 left) was developed as the first CCD type suitable for continuous (video) imaging. In an FT sensor, the exposure and readout functions of the sensor are physically separated with the exposure section built essentially identical to an FF sensor and a storage section almost a copy of the exposure section but covered with an opaque layer (usually aluminum). In an FT sensor, the exposure and readout can occur almost simultaneously because while the exposure section collects light for a new image, the previous image, held in the storage section, is shifted out.

The FT sensor is slightly more complicated to operate that an FF sensor because the exposure and storage section need separate shift drivers. In a typical cycle for a simple 30 frame-per-second progressive scan video camera, the exposure section is collecting light and the charge is not moving while the previous information in the storage section is shifting down to the readout section line by line. After the shifting is complete, the two sections are connected to the same clocks and the entire image is quickly moved down from the exposure section to the storage section. The line shift rate during the transfer is typically several hundred times faster than the readout line shifting rate. Since the charge pattern moves down very rapidly, the vertical charge smearing is reduced (but not eliminated) relative to the FF case. After the shift, the scanning is disconnected from the exposure section and the readout resumes at a slower shift rate. Due to the high speed of the transfer, no mechanical shutter is required / used.

Generally, the CCD cells in the storage section are smaller than the pixels in the exposure section because there is no need to efficiently collect light, no need to provide a square matrix and no need to provide anti-blooming protection (protection against cross-photosensitive cell charge shift for high energetic light points – blooming occurs when the charge in a pixel exceeds the saturation level and the charge starts to fill adjacent pixels, see Fig. 29). A few extra rows are usually added at the top of the storage section and covered with the opaque layer to provide a buffer between the edge of the light-collecting pixels and the storage cells. This prevents stray light from getting into the first few rows of

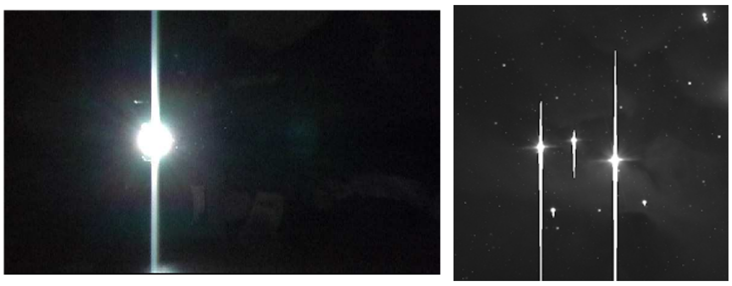


Figure 29: Smearing and blooming effects in CCDs configurations.

the storage area that contain image data where it can generate spurious charge that can produce white background patches in the image.

FT devices have typically faster frame rates than FF devices and have the advantage of a high duty cycle i.e. the sensor is always collecting light. FTs have the sensitivity of the full frame device and do not need a mechanical shutter but are typically more expensive due to the larger sensor size needed to accommodate the frame storage region.

In Interline Transfer (IT, see Fig. 28 right) sensors, the exposure and storage sections are alternated column by column in the same area of the silicon. Each column of photosensitive elements has next to it a vertical transfer register covered with aluminum. The group of transfer registers makes up the storage area. Light is collected in the photoelements, and then the accumulated charge is moved from all photoelements simultaneously into the neighboring transfer registers. After this move, the charge is shifted vertically, one line at a time, into the output register for readout.

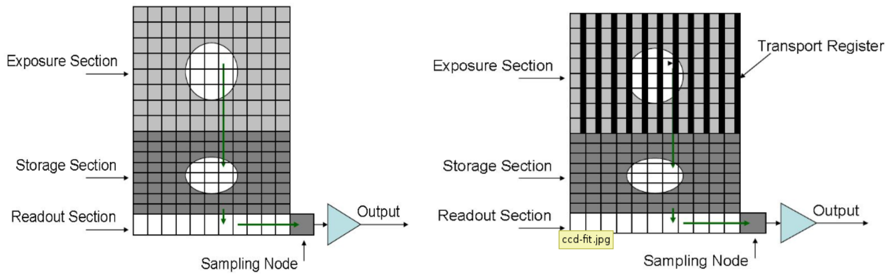


Figure 30: FT and FIT CCD configurations.

Although the photoelements appear to be in columns as in FT sensors, the photoelements in IT sensors are not connected together vertically because there is no need to shift vertically in the photosensitive areas. The very rapid image acquisition virtually eliminates image smear, at least for long exposure times. Additionally, because the photoelements are all isolated vertically from one another, the possibility of blooming along the photoelement columns is essentially eliminated. Antiblooming drains can be added between each column of photoelements and the transfer register for the column to the left as part of the isolation structure required between them. When the photoelements are isolated on all sides like this, blooming can be kept under very tight control. Altering the voltages at the photodiode so that the generated charges are injected into the substrate, rather than shifted to the transfer channels, can electronically shutter interline-transfer CCDs.

Interline devices have the disadvantage that the interline mask effectively reduces the light sensitive area of the sensor. This can be partially compensated by the use of microlens arrays to increase the photodiode fill factor (see Fig. 31). The compensation usually works best for parallel light illumination but for some applications which need wide angle illumination the sensitivity is significantly compromised.

The Frame Interline Transfer approach (FIT, see Fig. 30 right) was developed to provide both very low vertical smearing and electronic exposure control. Smearing in FT sensors, which have no exposure control, can be reduced only so far as the image can be moved rapidly from the imaging section to the storage section while IT sensors, which do have exposure control, have more smearing as the exposure time is reduced (the relation between exposure time and transfer time gets problematic). The FIT sensor is a combination of IT and FT in which the exposure section of an FT sensor is replaced by an IT sensor array. After the exposure, the charge can be shifted to the vertical transfer registers under the aluminum shields as in an IT device, but is then immediately shifted at high speed into an FT storage array below. As a result, the time available for accumulation of unwanted charge from light leaking under the shields is reduced to typically less than 1 millisecond with any exposure setting. The smearing is thus reduced by both the effect of the shields and the rapid transfer out of the exposure section. The disadvantage is of course the high cost due to the large sensor area and the reduction of the sensor area by analogy to the IT case.

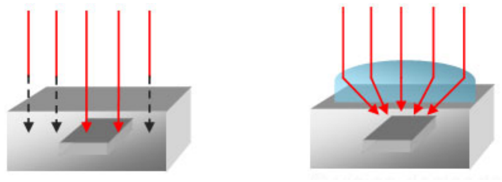


Figure 31: Microlens principle.

In both CMOS and CCD all photosensitive cell are sensitive to visible light, detect only brightness, not color. How to sense colour then ? There are different approaches (see Fig. 32):

- Colour Filter Array (CFA): Spatial multiplexing – sensors are made sensitive to red, green or blue using a filter coating that blocks the complementary light (note that this is similar to human cones !).
- 3 detectors: Foveon system using three layer sensors.
- Take 3 shots: temporal multiplexing or three different sensors.

Of course, the approach using three different sensors (usually CCD, see Fig. 34) avoids spatial multiplexing, but the camera needs to be bulky and therefore

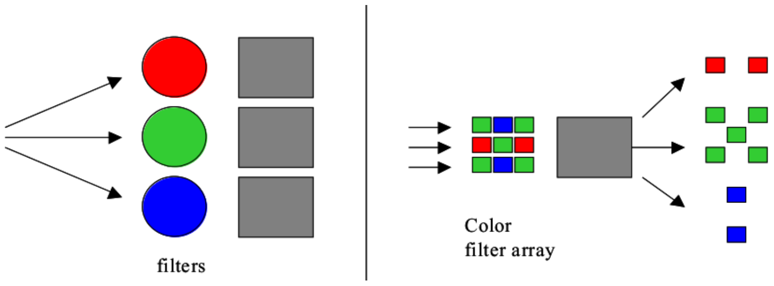


Figure 32: Approaches to colour sensing.

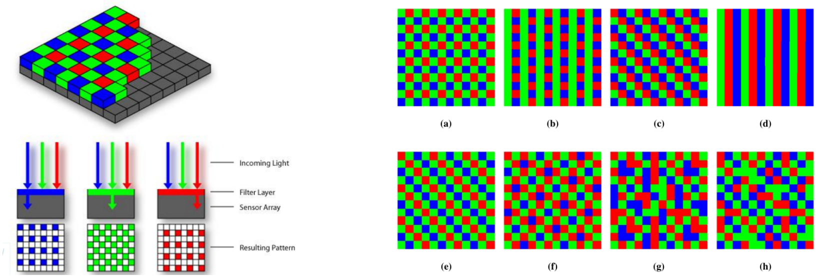


Figure 33: CFA examples.

gets expensive. Additionally, the amount of light reaching the sensor is reduced. Nikon Dichrioc is an example, which uses a microlense on top of a triplet of photoreceptors. Using dichroic filters wavelengths of light are separated to reach specific photoreceptors which record red, green, and blue wavelengths.

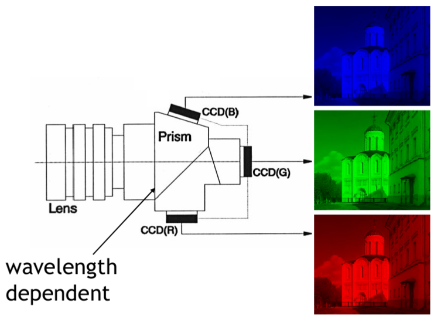


Figure 34: Non-spatial multiplexing colour sensing (multichip system.

Foveon X3 (see Fig. 35) is used in Sigma cameras and based on CMOS technology. There are three layers of photodiodes, silicon absorbs different “colors” at different depth, each layer captures a different color. No spatial multiplexing required.

So far, camera development has really followed the human visual system quite closely. If in the area of developing vehicules or means of transportation we had followed the same strategy, we had no cars, no bikes, no planes, no ships,

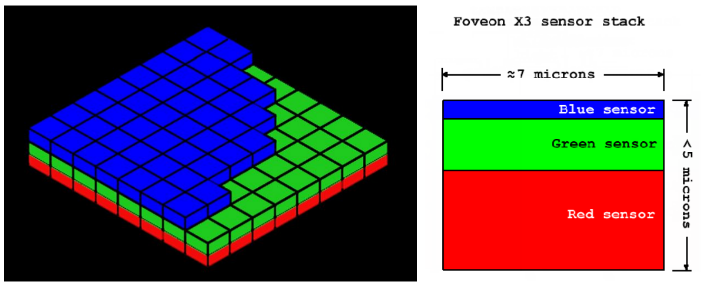


Figure 35: Non-spatial multiplexing colour sensing (Foveon system).

no spacecrafts but bipedal robots as in Fig. 36. As we shall see later, luckily imaging modalities have advanced beyond classical digital still image cameras (stereo vision, video, multiview video, range scanners like LIDAR / time of flight, structured light cameras, etc.).

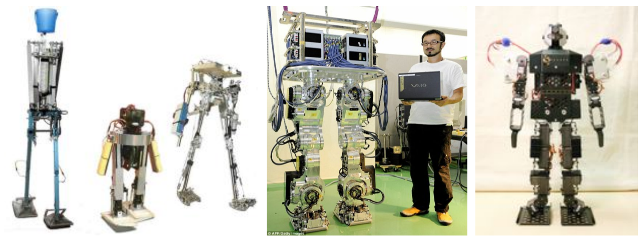


Figure 36: Means of transportation following the classical digital camera approach.

Fucussing done in cameras is fundamentally different as compared to the HVS – the lense does not change its shape (due to restrictive material properties) but cameras change the distance between lens and sensor surface to adjust for different object distances, something we cannot accomplish with our physiology. This can be done manually (eventually controlled by the HVS depending on the type of camera) or automatically (i.e. autofocus systems, see Chapter "Image Formation").

## 2.3 Image Properties

A signal is a function depending on some variable with physical meaning. Signals can be

- one-dimensional (e.g. dependent on time),
- two-dimensional (e.g. images dependent on two co-ordinates in a plane),
- three-dimensional (e.g. describing an object in space),
- or higher-dimensional

A scalar function may be sufficient to describe a monochromatic image, while vector functions are to represent, for example, color images consisting of three component colors.

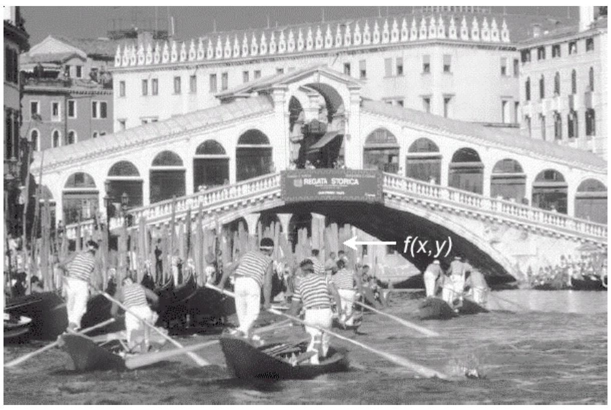


Figure 37: source image

Instead of an signal function a two-dimensional image matrix is used. Each element of the image matrix represents one pixel (*picture element*) with its position uniquely identified by the row- and column-index. The value of the matrix element represents the brightness value of the corresponding pixel within a discrete range.

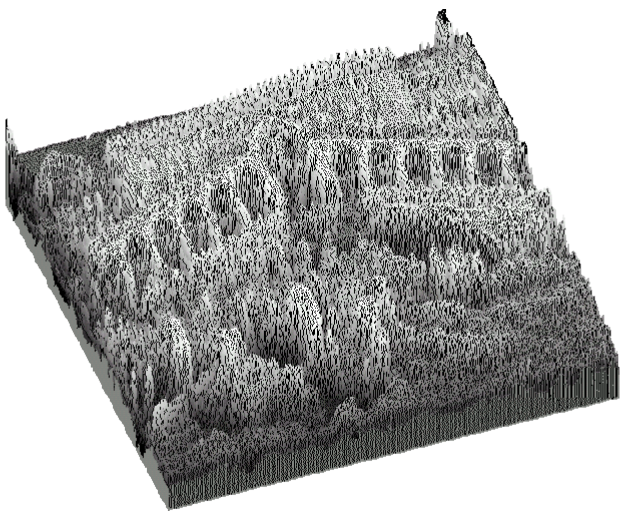


Figure 38: image as 2D-set of brightness values

### 2.3.1 Color Representation

In the RGB color modell (see figure 40) the luminance value is encoded in each color channel while in the YUV color modell (see figure 41) the luminance is only encoded in the Y channel.

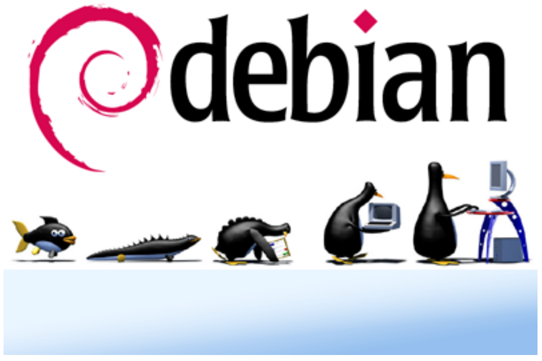


Figure 39: source image

$$Y \approx 0.5\text{Green} + 0.3\text{Red} + 0.2\text{Blue}$$

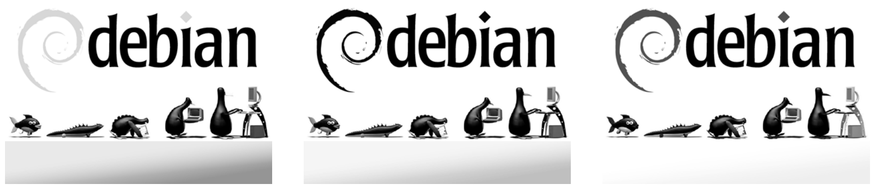


Figure 40: RGB decomposition of figure 39

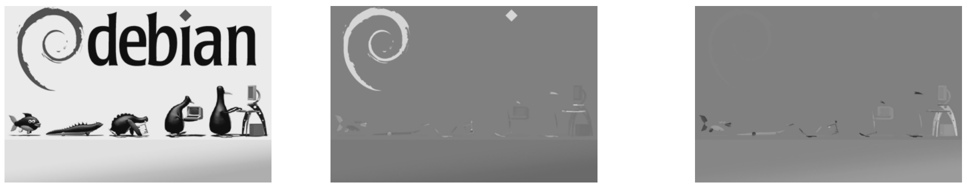


Figure 41: YUV decomposition of figure 39

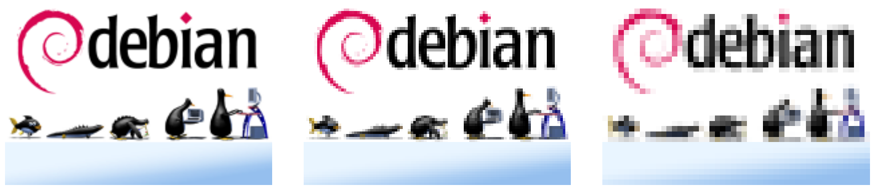
$$Y + U = Y + (B - Y) = Y - Y + B = B$$
$$Y + V = Y + (R - Y) = Y - Y + R = R$$
$$Y - B - R = (R + G + B) - B - R = G$$

**Palette** Images (i.e. "Malen nach Zahlen") include a lookuptable where an index identifies a certain color in unique way. Pixels do not carry luminance or color information but the index. Numerically close indices do not necessarily correspond to similar colors (e.g. .GIF).

### 2.3.2 Resolution and Quantisation

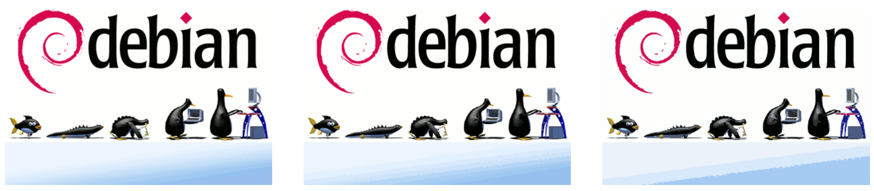
- Resolution (see figure 42) is the partition of the image into fixed segments (discretisation) - each segment is represented by a picutre element (pixel) which exhibits a constant luminance or colour value (resolution is the number of pixels used to represent the image content in both image dimensions).
- Quantisation (see figure 43) is the discrete number of gray or colour values used to represent luminance or colour information in a pixel (Bit per Pixel bpp).
- Storage requirements for an image is given as $b = N \cdot M \cdot \ln(F)$.
- The quality and perceivable detail of a digital image depends on the parameters $N$, $M$ und $F$ ab.

with $N$, $M$ ... height and width of the image given in pixels; $F$ ... numer of luminance values or colour values per pixel.



raster 1/2 raster 1/4 raster 1/8

Figure 42: Resolution of figure 39



256 colors 64 colors 16 colors

Figure 43: Quantisation of figure 39

### 2.3.3 Metric properites of digital images

The **distance** between two pixels in a digital image is a significant quantitative measure.
The distance between points with co-ordinates $(i, j)$ and $(h, k)$ may be defined in several different ways ...

**euclidean distance** $D_E((i, j), (h, k)) = \sqrt{(i-h)^2 + (j-k)^2}$

**city block distance** $D_4((i, j), (h, k)) = |i-h| + |j-k|$

**chessboard distance** $D_8((i,j),(h,k)) = \max\{|i-h|, |j-k|\}$

Pixel adjacency is another important concept in digital images. You can either have a 4-neighborhood or a 8-neighborhood as depicted in figure 44.

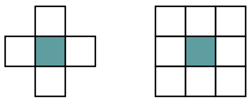


Figure 44: pixel neighborhoods

It will become necessary to consider important sets consisting of several adjacent pixels. So we define a **regions** as a contiguous set (of adjacent pixels).

There exist various **contiguity paradoxes** of the square grid as shown in figure 45.

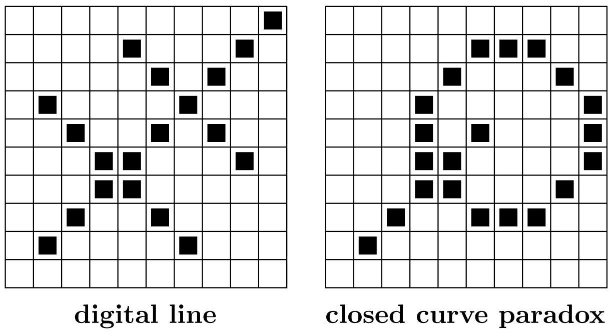


Figure 45: contiguity paradoxes

One possible solution to contiguity paradoxes is to treat objects using 4-neighborhood and background using 8-neighborhood (or vice versa). A hexagonal grid (as depicted in figure 46) solves many problems of the square grids. Any point in the hexagonal raster has the same distance to all its six neighbors.

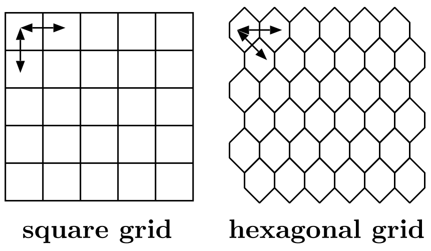


Figure 46: grid types

**Border** R is the set of pixels within the region that have one or more neighbors outside R. We distinguish between **inner** and **outer** borders.

An **edge** is a local property of a pixel and its immediate neighborhood – it is a vector given by a magnitude and direction. The edge direction is perpendicular to the gradient direction which points in the direction of image function growth.

Four **crack edges** are attached to each pixel, which are defined by its relation to its 4-neighbors as depicted in figure 47. The direction of the crack edge is that of increasing brightness, and is a multiple of 90 degrees, while its magnitude is the absolute difference between the brightness of the relevant pair of pixels.

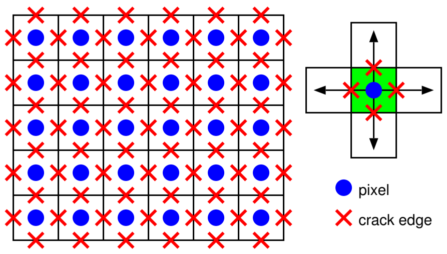


Figure 47: crack edges

The border is a global concept related to a region, while edge expresses local properties of an image function.

### 2.3.4 Histograms

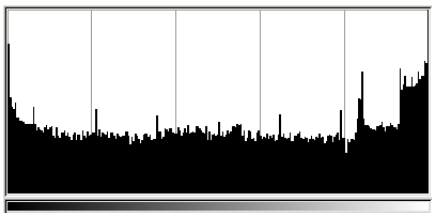


Figure 48: image histogram for figure 39

**Brightness histogram** provides the frequency of the brightness value $z$ in an image. Figure 48 shows the brightness histogram of the image in figure 39. Histograms lack the positional aspect as depicted in figure 49.

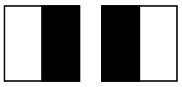


Figure 49: images with same histogram

However, histograms offer interesting properties, like *rotation-invariance* as well as *shift-invariance*, which means that the histogram does not change if the image is rotated or shifted (e.g. a circular wrapping of the entre image). These properties are of major importance in case of object detection or tracking.

## 2.4 Image Representation

**iconic images** consists of images containing original data; integer matrices with data about pixel brightness.

E.g., outputs of pre-processing operations (e.g., filtration or edge sharpening) used for highlighting some aspects of the image important for further treatment.

**segmented images** Parts of the image are joined into groups that probably belong to the same objects. It is useful to know something about the application domain while doing image segmentation; it is then easier to deal with noise and other problems associated with erroneous image data.

**geometric representations** hold knowledge about 2D and 3D shapes. The quantification of a shape is very difficult but very important.

**relational models** give the ability to treat data more efficiently and at a higher level of abstraction. A priori knowledge about the case being solved is usually used in processing of this kind.
Example: counting planes standing at an airport using satellite images ...
A priori knowledge

- position of the airport (e.g., from a map)
- relations to other objects in the image (e.g., to roads, lakes, urban areas)
- geometric models of planes for which we are searching
- ...

## 2.5 Traditional Data Structures

Some of the tradition data structures used for images are

- matrices,
- chains,
- graphs,
- lists of object properties,
- relational databases,
- ...

### 2.5.1 Matrices

Most common data structure for low level image representation Elements of the matrix are integer numbers. Image data of this kind are usually the direct output of the image capturing device, e.g., a scanner.

### 2.5.2 Chains

Chains are used for description of object borders. Symbols in a chain usually correspond to the neighborhood of primitives in the image.
The chain code for the example in figure 50 starting at the red marked pixel is:

000077665555555660000064444444422211111122344445652211

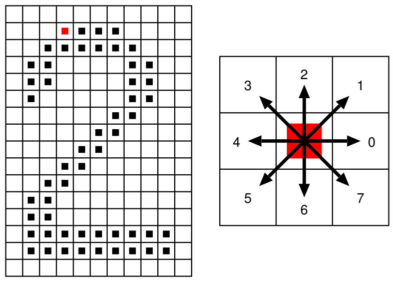


Figure 50: chain data structure

If local information is needed from the chain code, it is necessary to search through the whole chain systematically.

- Does the border turn somewhere to the left by 90 degrees?
- A sample pair of symbols in the chain must be found - simple.

If global information is needed, the situation is much more difficult. For example questions about the shape of the border represented by chain codes are not trivial.

Chains can be represented using static data structures (e.g., one-dimensional arrays). Their size is the longest length of the chain expected. Dynamic data structures are more advantageous to save memory.

### 2.5.3 Run length coding

not covered here.

### 2.5.4 Topological Data Structures

Topological data structures describe images as a set of **elements and their relations**.

This can be expressed in graphs (evaluated graphs, region adjacency graphs). For a region adjacency graph as an example of a topological data structure see figure 51.

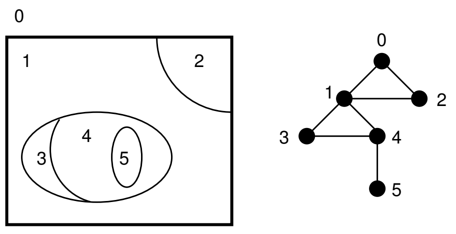
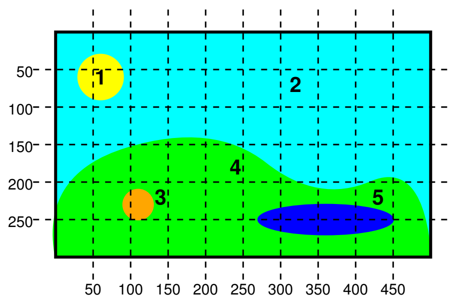


Figure 51: region adjacency graph

### 2.5.5 Relational Structures

Information is concentrated in relations between semantically important parts of the image: objects, that are the result of segmentation. For an example see figure 52.

This type of data structure is especially appropriate for higher level image understanding.



| no. | object name | colour | min. row | min. col. | inside |
|---|---|---|---|---|---|
| 1 | sun | yellow | 30 | 30 | 2 |
| 2 | sky | blue | 0 | 0 | – |
| 3 | ball | orange | 210 | 80 | 4 |
| 4 | hill | green | 140 | 0 | – |
| 5 | lake | blue | 225 | 275 | 4 |

Figure 52: relational data structure

## 2.6 Hierarchical Data Structures

Computer vision is by its nature very computationally expensive, if for no other reason than the amount of data to be processed. One of the solutions is using parallel computers (*brute force*). Many computer vision problems are difficult to divide among processors, or decompose in any way.

Hierarchical data structures make it possible to use algorithms which decide a strategy for processing on the basis of relatively small quantities of data. They work at the finest resolution only with those parts of the image for which it is necessary, using knowledge instead of brute force to ease and speed up the processing.

Two typical structures are **pyramids** and **quadtrees**.

Problems associated with hierarchical image representation are:

- Dependence on the position, orientation and relative size of objects.
- Two similar images with just very small differences can have very different pyramid or quadtree representations.
- Even two images depicting the same, slightly shifted scene, can have entirely different representations.

### 2.6.1 Pyramids

A **matrix pyramid** (M-pyramid) is a sequence $\{M_L, M_{L-1}, \ldots\}$ of images. $M_L$ has the same dimensions and elements as the original image. $M_{i-1}$ is derived from $M_i$ by reducing the resolution by one half (see figure 53). Therefore square matrices with dimensions equal to powers of two are required. $M_0$ corresponds to one pixel only.

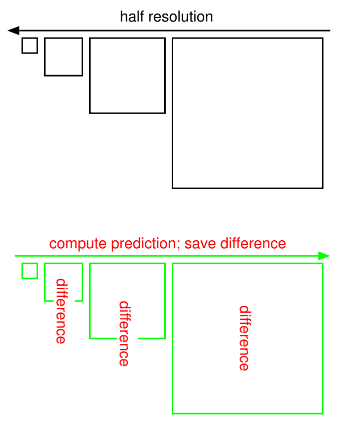


Figure 53: image pyramids

Only the difference between the prediction for $M_i$ (computed from $M_{i-1}$) and the real $M_i$ is saved for each level $i$ of the pyramid (see figure 53). Methods to reduce the image can be:

- subsampling
- averaging
- weighted averaging (Gauss'sche Bildpyramide)
- Laplace pyramide

Pyramids are used when it is necessary to work with an image at *different resolutions* simultaneously. An image having one degree smaller resolution in a pyramid contains four times less data, so that it is processed approximately four times as quickly.

A T-pyramid is a tree, where every node of the T-pyramid has 4 child nodes (as depicted in figure 54).

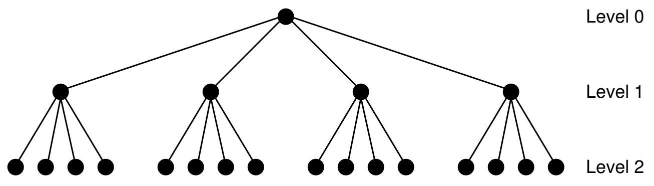


Figure 54: T-pyramid data structure

### 2.6.2 Quadtrees

Quadtrees are modifications of T-pyramids. Every node of the tree except the leaves has four children (NW: north-western, NE: north-eastern, SW: south-western, SE: south-eastern). Similarly to T-pyramids, the image is divided into four quadrants at each hierarchical level, however it is not necessary to keep nodes at all levels. If a parent node has four children of the same value (e.g., brightness), it is not necessary to record them. For an example refer to figure 55.

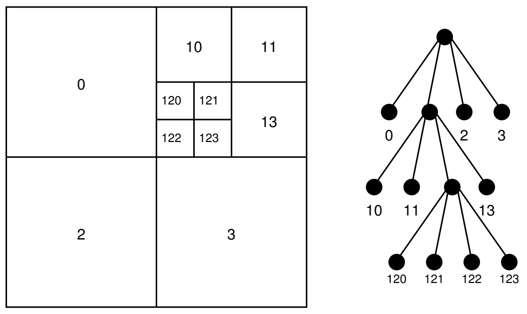


Figure 55: quadtree

## 2.7 Perception

As we have already observed, visual perception according to the Human Visual System (HVS) does not exactly correspond to the numerical values of single pixel but is highly context dependent.

**Definition 1 (Contrast)** *is the local change of luminance / intensity, defined das the fraction between the average object luminance and the average background luminance.*

Contrast is a logarithmic property. i.e. no linear relation between the numerical contrast values and the perceptual ones (in case of higher luminance we need more contrast to achieve the same impression as in low luminance conditions).

Acuity ("Sehschärfe") is best when luminance does not change too fast or too slow. In either cases HVS performance is much weaker and we face the problem of high inter-observer variability. Also, we have the "multiresolution property" caused by the non-uniform distibution of receptors in the retina.

# 3 Image Enhancement

The aim of Image Enhancement is to pre-process images in order to make them better suited for specific applications. These applications might include human viewing but this is not the most important one. More important is the preparation for subsequent image processing operations. We distinguish between:

- spatial domain methods (Bildraum)
- transform domain (e.g., frequency domain-Fourier, time-frequency domain-Wavelets)

## 3.1 Spatial Domain Methods

$f(x,y)$ is the intensity function of the original image and $g(x,y) = T(f(x,y))$ is the *enhanced* image. $T$ represents on operator applied to $f(x,y)$ in a specific neighbourhood[2] of $(x,y)$.

Simplest case: $1 \times 1$ neighbourhood, i.e. $g$ only depends on the value of $f$ at position $(x,y)$ ab. $T$ is called **grey-scale transformation** or transfer function (see figure 56) represented as $s = T(v)$ with $v = f(x,y)$ and $s = g(x,y)$

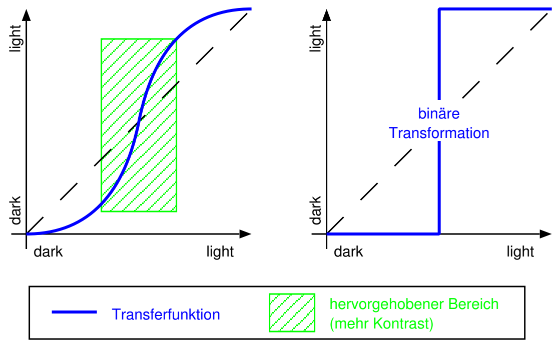


Figure 56: Grey-scale transformation

In the figure, the x-axis shows the original grey-scales, while the y-axis shows the values after transformation.

Largerneighbourhood: Different types of functions are often denoted as **masks**, templates, windows or filter. In most cases a small 2-D array (e.g. $3 \times 3$ pixel) is shifted across the image, computing the enhanced value at each pixel position.

[2]often, this is a squared image region centered in $(x,y)$. The center of this image region is moved from pixel to pixel.

The coefficients in the array are chosen as to emphasize or suppress certain image properties.

Example: Given an image with constant intensity with isolated pixels exhibiting different intensity ("pop noise"); we select the mask to be $w_i = -1\, i = 1, \ldots 9$ except for $w_5 = 8$, each entry of the mask is multiplied with the pixels positioned below the entry and all the results are added up. For an area of constant intensity we get 0 as a response, the mask is shifted across the image pixel by pixel (see figure 57).

$$\text{Result} \begin{cases} = 0 & \text{all pixels identical} \\ > 0 & \text{central pixel is larger than surrounding} \\ < 0 & \text{central pixel is smaller than surrounding} \end{cases}$$

| | ... | $x-1$ | $x$ | $x+1$ | ... |
|---|---|---|---|---|---|
| | ... | ... | ... | ... | ... |
| $y-1$ | ... | o | o | o | ... |
| $y$ | ... | o | x | o | ... |
| $y+1$ | ... | o | o | o | ... |
| | ... | ... | ... | ... | ... |

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

Figure 57: Image and mask

$$T[f(x,y)] = w_1 f(x-1,y-1) + w_2 f(x-1,y) + w_3 f(x-1,y+1) + \ldots + w_9 f(x+1,y+1)$$

## 3.2 Contrast Manipulation & Modification

### 3.2.1 Changing the Amplitude

Changing the range of grey-scales (see figure 58: Visualisation of difference images after prediction or MC; clipping is often used in case of a small number of pixels is found at the tails of the histogram - contrast is improved additionally).

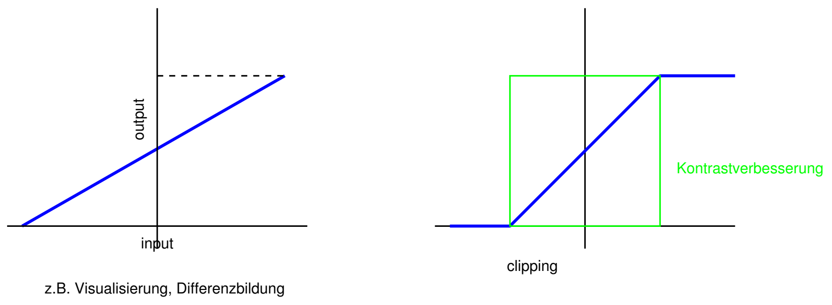


Figure 58: Modification of grey-scale range

Local vs. global: All techniques discussed here in this section can be applied to the entire image - globally - or to parts / tiles of the image - locally.

### 3.2.2 Contrast Modification

Contrast is increased in areas where the slope of the transfer function (or its tangent) is larger than 1.

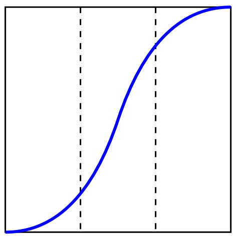


Figure 59: Contrast Modification

As an example, we show the contrast modification of a computer tomography by applying the logarith as transfer function in figure 60.

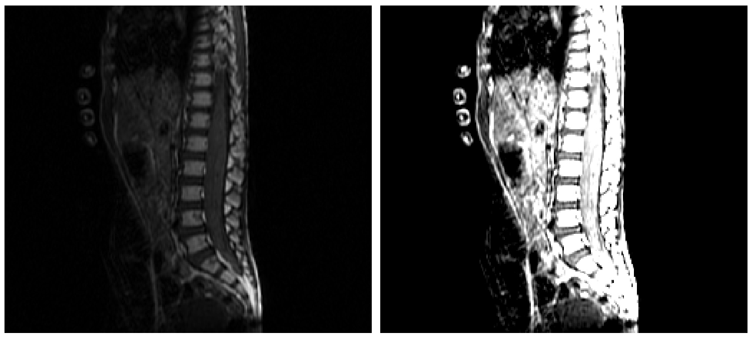


Figure 60: Contrast modification of a CT

Contrast can be modified by using simple transfer functions like

$$s = r^p \qquad p = 2, 3, 1/2$$

(see also figure 59). Further typical contrast modification techniques are shown in figure 61.

A further example is displayed applying the logarithm function to a Myelin image (similar $s = r^{1/2}$) in fig. 62).

### 3.2.3 Histogram Modification

Histogram: Distribution of relative frequency of grey-values in an image (global image description).

Let $r$ be the grey-value of a pixel and $0 \leq r \leq 1$ with $r = 0 =$ black and $r = 1 =$ white. We consider the transfer function $s = T(r)$ with the properties

(a) $T$ is monotonically increasing on $(0, 1)$

(b) $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$

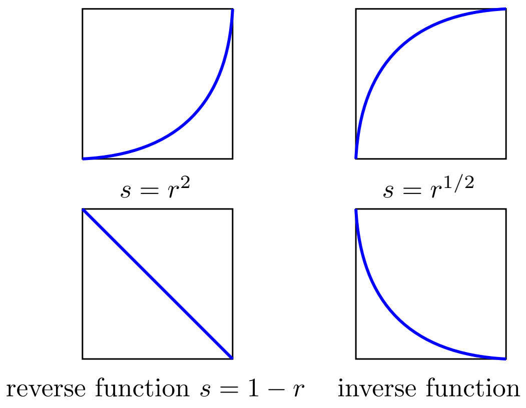


Figure 61: Typical contrast modification techniques

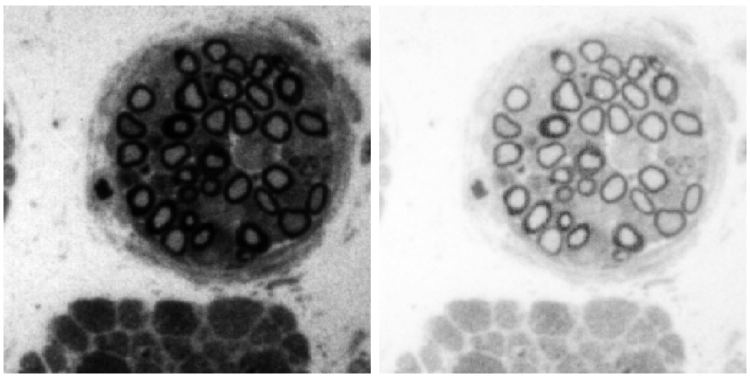


Figure 62: Contrast modification of a Myelin

The inverse transform from $s$ to $r$ is $r = T^{-1}(s)$ with $0 \leq s \leq 1$ with identical properties (a) und (b).

Consider the grey-values as being contineous random variables. We can represent the original and transformed grey-value distributions by considering their corresponding density functions $p_r(r)$ and $p_s(s)$. The density functions describe the overall impression of the image.

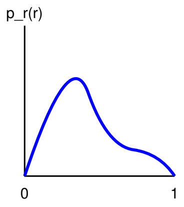


Figure 63: Contineous histogram

Remark: These density functions may be interpreted as a *contineous histogram.*

From elementary probability theory and statistics we know:
If we know $p_r(r)$, $T(r)$, and $T^{-1}(s)$ satisfies the condition (a), the density function of the transformed grey-values is given by:

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

### 3.2.4 Histogram-Equalisation

We consider the following transfer function (called cumulative distribution function):

$$s = T(r) = \int_0^r p_r(w)dw \qquad 0 \leq r \leq 1$$

When computing the derivative with respect to $r$ we result in (fundamental theorem of calculus):

$$\frac{ds}{dr} = p_r(r) \tag{2}$$

When inserting equation (2) into equation (1) we get:

$$p_s(s) = \left[ p_r(r) \frac{1}{p_r(r)} \right]_{r=T^{-1}(s)} = 1 \;\; 0 \leq s \leq 1 \,. \tag{3}$$

The result is a uniform density, constant 1. This result is entirely independent of the inverse function. Funktion.

Histogram equalisation is achieved by applying the cumulative distribution function (CDF) as grey-value transfer function. In the equalised histogram all occurence probabilities are equal to 1 (attention: here we are in the – *idealised* – contineous case !. Fig. 65 shows examples of CDF and corresponding densities.

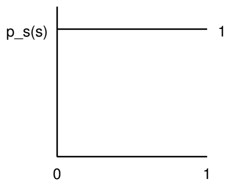


Figure 64: equalised histogram

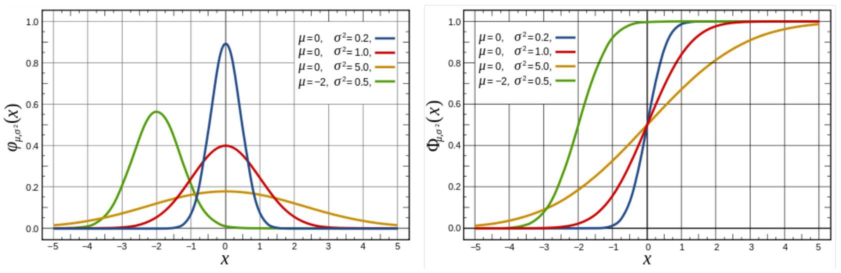


Figure 65: Gaussians and their cumulative distribution functions

Example:

$$p_r(r) = \begin{cases} -2r+2 & 0 \le r \le 1 \\ 0 & \text{otherwise} \end{cases}$$

$$s = T(r) = \int_0^r (-2w+2)dw = -r^2 + 2r$$

$$r = T^{-1}(s) = 1 - \sqrt{1-s}$$

Problem: For natural images no “Function” $p_r(r)$ exists.

**Discretisation**

$$p_r(r_k) = \frac{n_k}{n} \qquad 0 \le r_k \le 1, \quad k = 0, 1, \dots, L-1$$

$n_k$ ... number of occurence of grey-scale $k$
$n$ ... number of pixel
$L$ ... number of grey-scales

**Discrete Equalisation**

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j) \qquad k = 0, \dots, L-1, \quad r_k = T^{-1}(s_k)$$

Remark: The inverse function is not required. $T(r_k)$ can be derived from pixel statistics. Due to discretisation the result is an approximation only.

An image enhanced by histogram equalisation is shown in figure 66. In addition to the original and the enhanced image the corresponding histograms are shown.

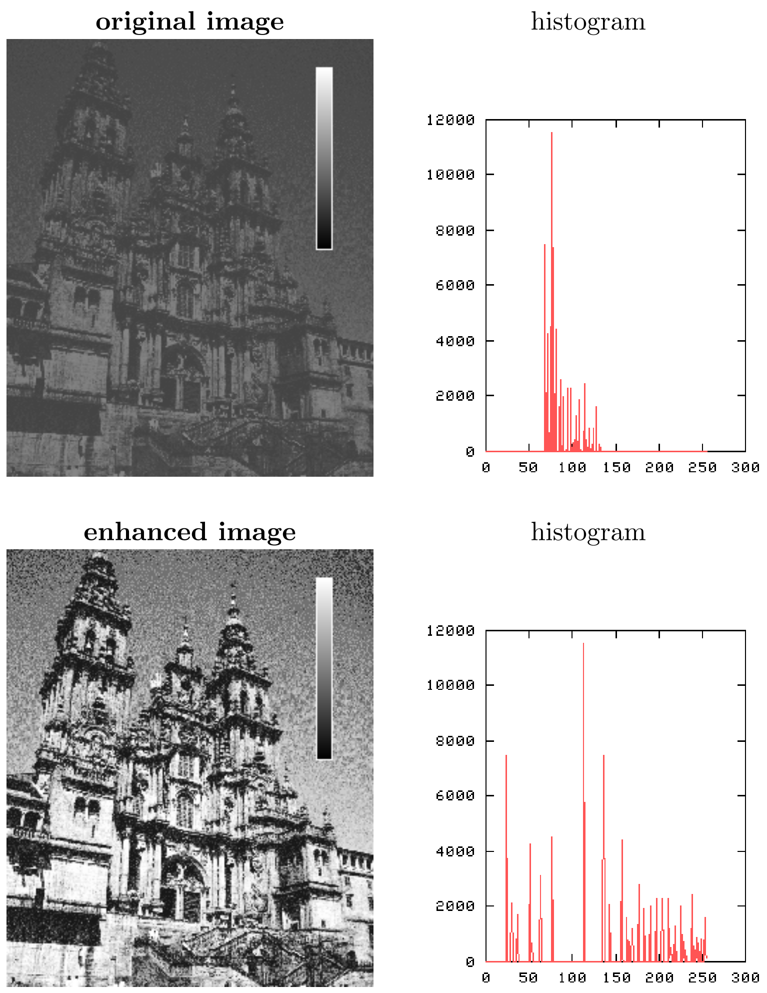


Figure 66: Histogram equalisation

### 3.2.5 Explicit Histogram Specification

Let $p_r(r)$ be the original and $p_z(z)$ the target density function. In the first step, the original image is histogram equalised:

$$s = T(r) = \int_0^r p_r(w)dw$$

Assuming the target image to be avaialable, it could be histogram equalised as well:

$$v = G(z) = \int_0^z p_z(w)dw$$

$z = G^{-1}(v)$ would result in the target pixel vaues.

$p_s(s)$ and $p_v(v)$ have identical uniform densities[3]. Therefore it is possible to use $s$ (from the equalised original) instead of $v$ in the inverse process. So $z = G^{-1}(s)$ exhibits the desired target density.

Procedure:

1. Equalise original image $\rightarrow s$
2. Specify the desired target density and obtain $G(z)$
3. $z = G^{-1}(s) \Rightarrow z = G^{-1}(T(r))$

Problem: The inverse function cannot be computed directly in the discrete case we have here. Thus, the inverse function is obtained by a mapping grey-scale to grey-scale (table lookup).

Application: Optimisation for specific output devices, for which the optimal target histogram is known, e.g. for large plotters etc.

Remark: The techniques discussed so far can also be applied to $n \times m$ neighbourhoods – in case it is only a specific region which is of interest, this leads to better results. As an example, we discuss contrast limited adaptive histogram equalisation (CLAHE). Classical, global histogram equalisation works well in case the distribution of the pixel values (i.e. the histogram) is similar throughout the image. In case the image contains areas which are significantly lighter or darker than the overall histogram suggests, the contrast of those regions will not be sufficiently enhanced.

Adaptive histogram equalisation (AHE) transforms each pixel with a tranformation function derived from the pixels neighbourhood (can be a fixed square, can be more involved, the computation may be weighted, etc.) – again, the CDF computed from pixels in the neighbourhood is used. In case the neighbourhood is a very homogeneous area, the histogram will be very peaked and the transformation function will map a narrow range of pixel values to the whole range of the result image, resulting in an over-amplification of **noise**.

This is where CLAHE comes in: The contrast is limited in each neighbourhood – since the slope of the transformation function determines the contrast amplification and this slope is proportional to the slope of the CDF (which is locally

[3]gleichmäßige Dichte

proportional to the histogram value of the pixel), the histogram is clipped at some predefined histogram value as shown in Fig. 67. This limits the slope of theof contrast enhance CDF and thus, the amount of contrast enhanceent. The clip-value of often chosen to be 3 times the grey mean value. Due to intensitiy / luminance loss, it is better to redistribute the lost parts to the other histogram bins.

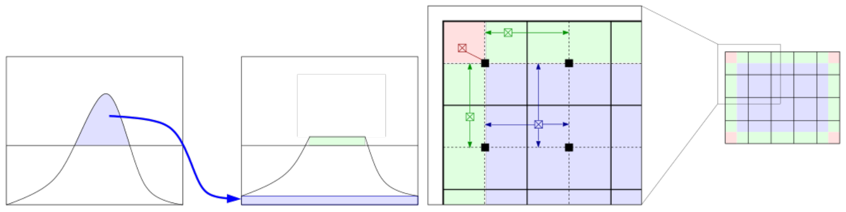


Figure 67: CLAHE: histogram clipping and interpolation

Since the computation of CLAHE involves the determination of the transformation function at each pixel, it is usually approximated only, by computing transformation functions for fixed tiles of an image grid. The actual output pixel for a specific location is then computed using up to four transformation functions and appropriate bilinear or linear interpolation techniques as shown in the Fig. 67.

## 3.3 Image smoothing & Denoising

Aim: Effects caused by transmission errors or sampling errors should be corrected. These effects are **local errors** (in the ideal casse single independent pixel)!

The most popular technique is *Neighbourhood Averaging* described in the following.

### 3.3.1 Neighbourhood Averaging

$g(x, y)$ is obtained by computing averages in a neighbourhood $S$:

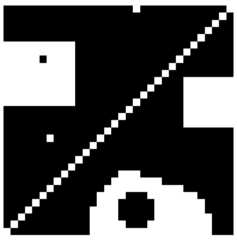
$$g(x, y) = \frac{1}{M} \sum_{(n,m) \in S} f(n, m)$$

$M$ ... number of pixels in $S$.

In neighbourhood averaging masks like shown in figure 57 are used for defining the neighbourhood:

Problem: Edges get significantly softened (*blurring*)! This effect can be handled by applying thresholding (with threshold $T$). If the difference between original and "enhanced" pixel value is too large, averaging is avoided and the origianl vaulue is set.

$$\hat{g}(x, y) = \begin{cases} g(x, y) & |f(x, y) - g(x, y)| < T \\ f(x, y) & \text{sonst} \end{cases}$$



$$\rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \rightarrow$$

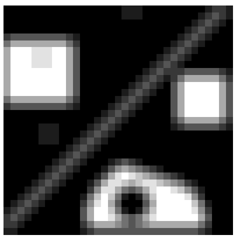


Figure 68: Averaging with $3 \times 3$ maske

### 3.3.2 Median Filtering

Instead of computing an average in the neighbourhood a median is used instead. Statistical outlyers in the neighbourhood are not included in the generation of the enhanced pixel value. Especially for denoising (e.g. pop noise) the median-based approach is often preferable (see Fig. 69).

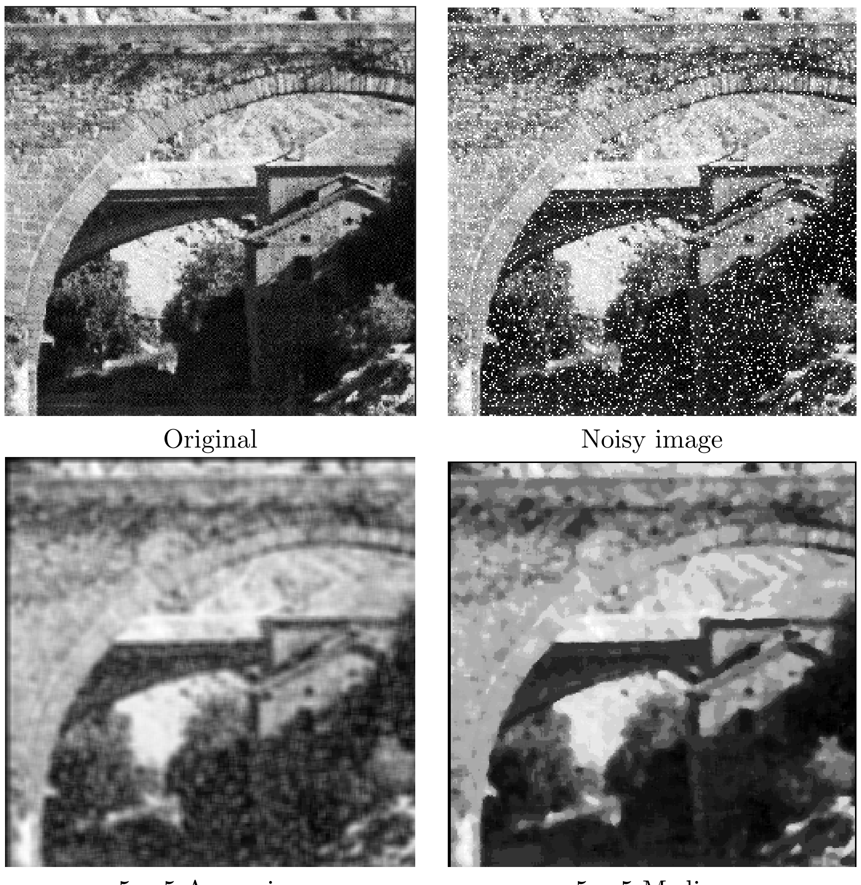


Figure 69: Denoising

## 3.4 Image Sharpening

The aim of *image sharpening* is to emphasize edges;
Idea: Difference among pixels suggests the existence of an edge.

Averaging and sharpening are based on two antagonistic mathematical concepts: While in averaging, detail are "integrated", sharpening "differentiates" details.

Gradient $G[f(x,y)] = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$

1. $G$ points into the direction of the largest growth of $f(x,y)$
2. $|G[f(x,y)]| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \sim \text{mag}(G)$

mag$(G)$ ... magnitude of $f$, is equal to the largest growth rate of $f(x,y)$.

In image processing mag$(G)$ is often denoted as Gradient for simplicity.

**Discretisation**
Derivatives are approximated by differences:

$$|G[f(x,y)]| = \sqrt{[f(x,y) - f(x+1,y)]^2 + [f(x,y) - f(x,y+1)]^2}$$

As an alternative, absolute values can be used instead of the square root (more efficient implementation).

**Roberts Operator** (see also chapter 5.1.1) and Fig. 70).

$$|G[f(x,y)]| = \max\{|f(x,y) - f(x+1,y+1)|, |f(x+1,y) - f(x,y+1)|\}$$

Overall, the value of the Gradient is proportional to the difference among pixels grey-values – large vaues for edges, small values for smooth or uniform areas.

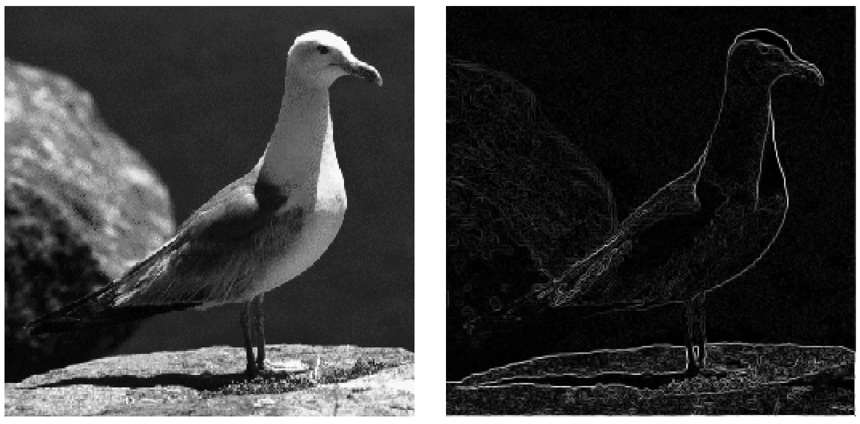

Original Roberts Gradient image

Figure 70: Image Sharpening

There are several possibilities how to visualise the *Gradient image* $g(x,y) = |G[f(x,y)]|$:

| | |
|---|---|
| $g(x,y) = G[f(x,y)]$ | $g(x,y) = \begin{cases} G[f(x,y)] & G \geq S \\ f(x,y) & \text{sonst} \end{cases}$ |
| $g(x,y) = \begin{cases} T_{\text{const}} & G \geq S \\ f(x,y) & \text{sonst} \end{cases}$ | $g(x,y) = \begin{cases} T_1 & G \geq S \\ T_2 & \text{sonst} \end{cases}$ |

Figure 71: Types of Gradient visualisation

## 3.5 Transformation-based Techniques

Transformations used in image processing are unitary[4] transformations and are used for:

**Feature extraction** to describe certain properties in an efficient manner (e.g. frequencies: high - edges, low - luminance). The aim is to be able to conduct certain operations more efficiently in the transformed domain (e.g. denoising).

**Compression** concentration of information

**Efficient calculations** e.g. a *dense matrix* is transformed into a *sparse matrix*, since more efficient algorithms do exist for sparse martices (in sparse matrices – sparsely populated matrices – many coefficients are equal to zero).

In many cases the concept to represent a signal using orthogonal basis functions is used.

Background: Vectors in 2 dimensional space can be represented by a set of orthogonal (i.e. the inner product is zero) basis-vectors (orthogonal basis):

$$(x, y) = \alpha(1, 0) + \beta(0, 1).$$

$\{(1,0),(0,1)\}$ are the orthogonal basis-vectors. $\alpha$ and $\beta$ are the coefficients which determine the weight of each basis-vector to represent the vector $(x, y)$. The orthogonality of the vectors facilitates a minimal number of basis-vectors.

This concept can be generalised to functions and signals, respectively.

$$f(x) = \sum_n < f(x), \psi_n(x) > \psi_n(x)$$

Functions $\psi_n(x)$ are orthogonal basis functions, $< f(x), \psi_n(x) >$ are the transform coefficients which determine the weight of each basis function to represent a given signal "well". For an application the coefficients $< f(x), \psi_n(x) >$ are computed and processed further. Since the basis functions $\psi_n(x)$ are orthogonal, the required number to represent the signal is minimal.

e.g.: In case of the Fourier transform (see below) the basis functions are $\psi_n(x) = e^{-\pi inx} = \cos(nx) - i\sin(nx)$. Here, frequencies of periodic signals are considered. A Fourier coefficient $< f(x), \psi_n(x) >$ represents the strength / energy of the frequency $n$ in a signal. Obviously, not all signals may be represented efficiently using this approach.

[4]orthogonal and regular

### 3.5.1 Fourier Transform

Developed by French Fourier who was very interested in music (violin) and wanted to know how sounds are created by changing the length of the cords. For more background, see e.g. `http://de.wikipedia.org/wiki/Fourier-Transformation` .

Let $f(x)$ be a contineous function, $\hat{f}(u)$ is the Fourier transform of $f(x)$ with respect to frequency $u$.

$$\hat{f}(u) = \int_{-\infty}^{\infty} f(x) e^{-2\pi iux} dx \tag{4}$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(u) e^{2\pi iux} du \tag{5}$$

The inversion can be computed in case $f(x)$ is contineous and can be integrated and in case $\hat{f}(u)$ can be integrated as well.

The Fourier transform of a real function is usually of complex values.

$$\hat{f}(u) = \Re(u) + i\Im(u)$$

$$\hat{f}(u) = |\hat{f}(u)| e^{i\Phi(u)}$$

$$|\hat{f}(u)| = \sqrt{\Re^2(u) + \Im^2(u)} \qquad \Phi(u) = \tan^{-1}\left(\frac{\Im(u)}{\Re(u)}\right)$$

| | |
|---|---|
| $\lvert\hat{f}(u)\rvert^2$ | ... Power-Spectrum (Spektraldichte) |
| $\lvert\hat{f}(u)\rvert$ | ... Fourier-Spectrum (Frequenzspektrum) |
| $\Phi(u)$ | ... Phase angle |
| $u$ | ... Frequency variable (since $e^{2\pi iux} = \cos 2\pi ux + i \sin 2\pi ux$) |

If we interpret the integral as the summation of discrete terms, it gets clear that $\hat{f}(u)$ is composed of an infinite sum of Sine- and Cosine terms, where the parameter $u$ determines the frequency of the Sine/Cosine pair.

**Discrete Fourier Transform (DFT)** $\{f(0), f(1), \ldots, f(N-1)\}$ are $N$ uniformly sampled points of a contineous function. The DFT is defined as

$$\hat{f}(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-2\pi iux} \qquad u = 0, \ldots, N-1 \tag{6}$$

$$f(x) = \sum_{u=0}^{N-1} \hat{f}(u) e^{2\pi iux/N} \qquad x = 0, \ldots, N-1 \tag{7}$$

**Two-dimensional**

$$\hat{f}(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i(ux/M + vy/N)} \tag{8}$$

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \hat{f}(u,v) e^{2\pi i(ux/M + vy/N)} \tag{9}$$

For Display-purposes it is better to use $D(u,v) = \log(1+|\hat{f}(u,v)|)$ than $|\hat{f}(u,v)|$, since values decrease rapidly for increasing frequency.

Examples of DFT transforms are visualised in figures 72 and 73.

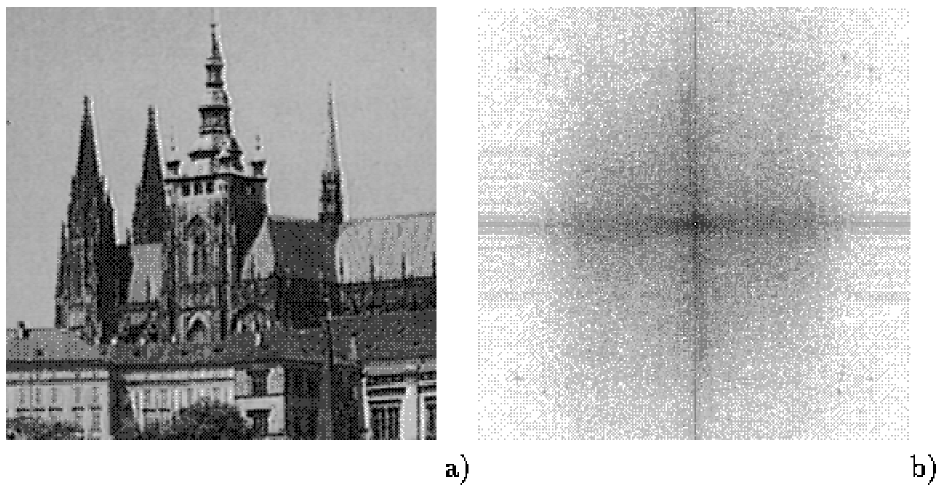


Figure 72: Origial image and its Fourier Spectrum (Magnitude)

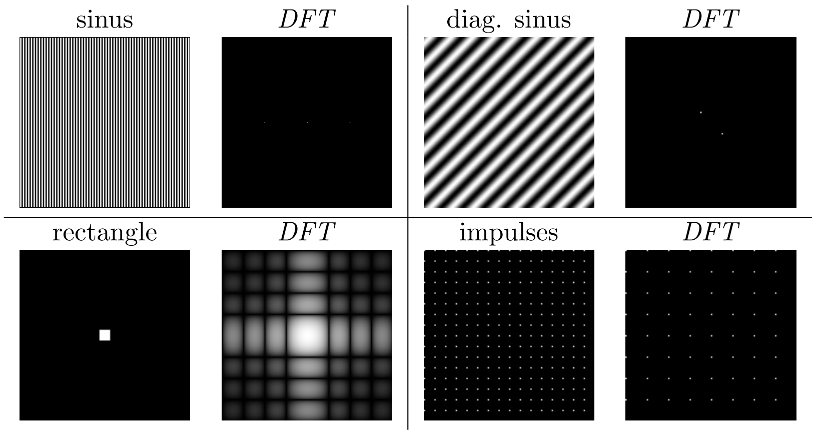


Figure 73: DFT Transformations

**Properties of the 2D Fourier Transform**

- $\hat{f}(0,0)$ is identical to the average grey-value of all pixels

$$\hat{f}(0,0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

- **Separability**

$$\hat{f}(u,v) = \frac{1}{M}\sum_{x=0}^{M-1}\left(\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi ivy/N}\right)e^{-2\pi iux/M}$$
$$f(x,y) = \sum_{u=0}^{M-1}\left(\sum_{v=0}^{N-1}\hat{f}(u,v)e^{2\pi ivy/N}\right)e^{2\pi iux/M}$$

  This means that the two dimensional transform my be implemented as a consecutive conduct of two one-dimensional transforms, i.e. applying a DFT to all rows and subsequentl to all columns (or vice versa). The foundation of this property is the separability of the underlying basis functions, i.e. $e^{-2\pi i(ux+vy)} = e^{-2\pi iux}e^{-2\pi ivy}$.

- **Translation**

$$\hat{f}(u-u_0, v-v_0) = f(x,y)e^{2i\pi(u_0x/M+v_0y/N)} \tag{10}$$
$$f(x-x_0, y-y_0) = \hat{f}(u,v)e^{-2i\pi(ux_0/M+vy_0/N)} \tag{11}$$

  Set $u_0 = M/2$ and $v_0 = N/2$

$$\hat{f}(u-M/2, v-N/2) = f(x,y)e^{i\pi(x+y)} = (-1)^{x+y}f(x,y)$$

  The origin of the Fourier Transform $(0,0)$ can be moved to the conter of the frequency plane $(M/2, N/2)$ by multiplying $f(x,y)$ with $(-1)^{x+y}$ **and** a shift in $f(x,y)$ does not affect $|\hat{f}(u,v)|$ (shift invariance of the DFT).

$$|\hat{f}(u,v)e^{-2\pi i(ux_0/M+vy_0/N)}| = |\hat{f}(u,v)|$$

- **Periodiosity**

$$\hat{f}(u,v) = \hat{f}(u+N,v) = \hat{f}(u,v+M) = \hat{f}(u+aN, v+bM)$$

- **Symmety** In case $f(x,y)$ real-valued:

$$\hat{f}(u,v) = \hat{f}^*(-u,-v) \qquad\qquad |\hat{f}(u,v)| = |\hat{f}(-u,-v)|$$

  Caused by the conjugate symmetry property around the origin, half of the transform coefficients are redundant. Symmetry and periodicity facilitate to keep the entire period and to shift the origin of the transform domain into $(M/2, N/2)$ as described before.

- **Linear combination**

$$k_1f(x,y) + k_2g(x,y) \Leftrightarrow k_1\hat{f}(u,v) + k_2\hat{g}(u,v)$$

- **Scaling**

$$af(x,y) = a\hat{f}(u,v) \text{ contrasting to } \qquad f(ax,by) = \frac{1}{ab}\hat{f}(u/a, v/b)$$

Scaling can be shown as follows (1-dim.): $\hat{f}(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iux}dx$ for$f(x)$. For f(ax) we get by analogy $\hat{f}(u) = \int_{-\infty}^{\infty} f(ax)e^{-2\pi iux}dx$. Multiplication of the integral and the exponent by $a/a$ leads to $1/a \int_{-\infty}^{\infty} f(ax)e^{-2\pi iax(u/a)}adx$. Applying a substitution of variables $s = ax$ $(ds = adx)$ we result in $1/a \int_{-\infty}^{\infty} f(s)e^{-2\pi is(u/a)}ds$. This expression is evidently equal to $\frac{1}{a}\hat{f}(\frac{u}{a})$. A contracted function $(a > 1)$ consequently exhibits a Fourier transform with reduced amplitude and horizontal stretching in frequency space.

**Laplacian**

$$\nabla^2 f(x,y) = \frac{\partial f}{\partial x^2} + \frac{\partial f}{\partial y^2}$$
$$\widehat{\nabla^2 f(x,y)} = -(2\pi)^2(u^2+v^2)\hat{f}(u,v)$$

**Convolution** Convoluting the mask $h(x)$ with the image $f(x)$ is defined as

$$h(x) * f(x) = \int_{-\infty}^{\infty} h(\alpha)f(x-\alpha)d\alpha$$

**Convolution Theorem**

$$f(x) * g(x) \Leftrightarrow \hat{f}(u) \cdot \hat{g}(u) \tag{12}$$
$$f(x) \cdot g(x) \Leftrightarrow \hat{f}(u) * \hat{g}(u) \tag{13}$$

$f(x) * g(x)$ ... exhibits increasing computational complexity with increasing size of the mask $f$.
$\hat{f}(u) \cdot \hat{g}(u)$ ... no increasing complexity if mask $f$ is known

Thus, the convolution theorem is applied to ...

**Reduction of complexity of convolution** Fourier Transforms of $f$ and $g$ are computed and the results multiplied, the product is inverse Fourier transformed (pays off with a mask size larger than $20^2$ pixels)

**Filtering in Frequency space** (see chapter 3.5.2)

**Fast Fourier Transformation (FFT)** FFT was published in 1968 by Cooley and Tuckey but relies on an idea of C.F. Gauss in the area of matrix factorisation. The computational complexity of the DFT when applied to $N$ datapoints is $\mathcal{O}(N^2)$ which is too high even for todays advanced hardware. FFT reduces complexity to $\mathcal{O}(N \log N)$ and is the enabler of an application of Fourier techniques in signal processing.

### 3.5.2 Filtering in Frequency domain

$$g(x,y) = h(x,y) * f(x,y) \tag{14}$$
$$\hat{g}(u,v) = \hat{h}(u,v) \cdot \hat{f}(u,v) \tag{15}$$

$\hat{h}(u,v)$ ... Transfer function
$g(x,y)$ ... Shifting the mask $h(x,y)$ across the image $f(x,y)$

**Procedure** ($f(x,y)$ is given)

- Compute $\hat{f}(u,v)$
- choose $\hat{h}(u,v)$ in a way, that the resulting image emphasises certain properties
- Compute the enhanced image by applying the inverse Fourier transform to $\hat{h}(u,v) \cdot \hat{f}(u,v)$

In figure 74 shows the filters described in the following.

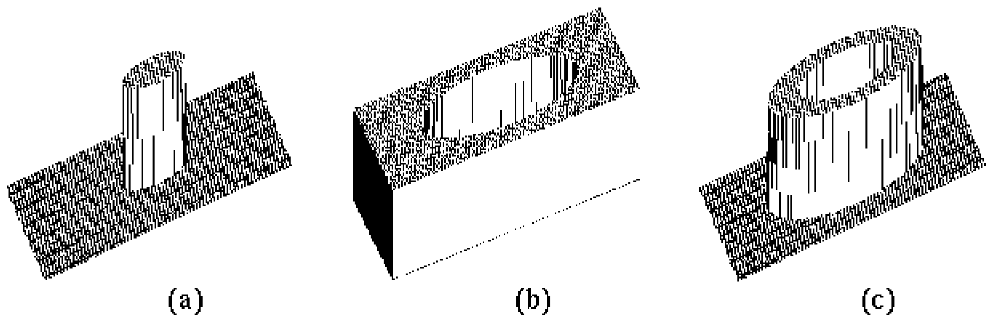


Figure 74: Different types of filters

**Lowpass Filter** Edges and shrp transitions are phenomena of high frequency nature. If these parts are surpressed in the frequency domain, the image gets smoothed.
$\hat{h}(u,v)$ is the *Ideal Lowpass Filter* (ILPF)

$$\hat{h}(u,v) = \begin{cases} 1 & D(u,v) \leq D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

$D_0$ is the so-called Cut-off Frequency and $D(u,v) = (u^2 + v^2)^{1/2}$ is the distance between $(u,v)$ and the origin. Applying $\hat{h}(u,v) \cdot \hat{f}(u,v)$ zeros a high frequency parts (edges), low frequency parts are retained (see Fig. 75). Filters of this type affect real- and imaginary parts but do not change the phase (*zero-phase shift*).

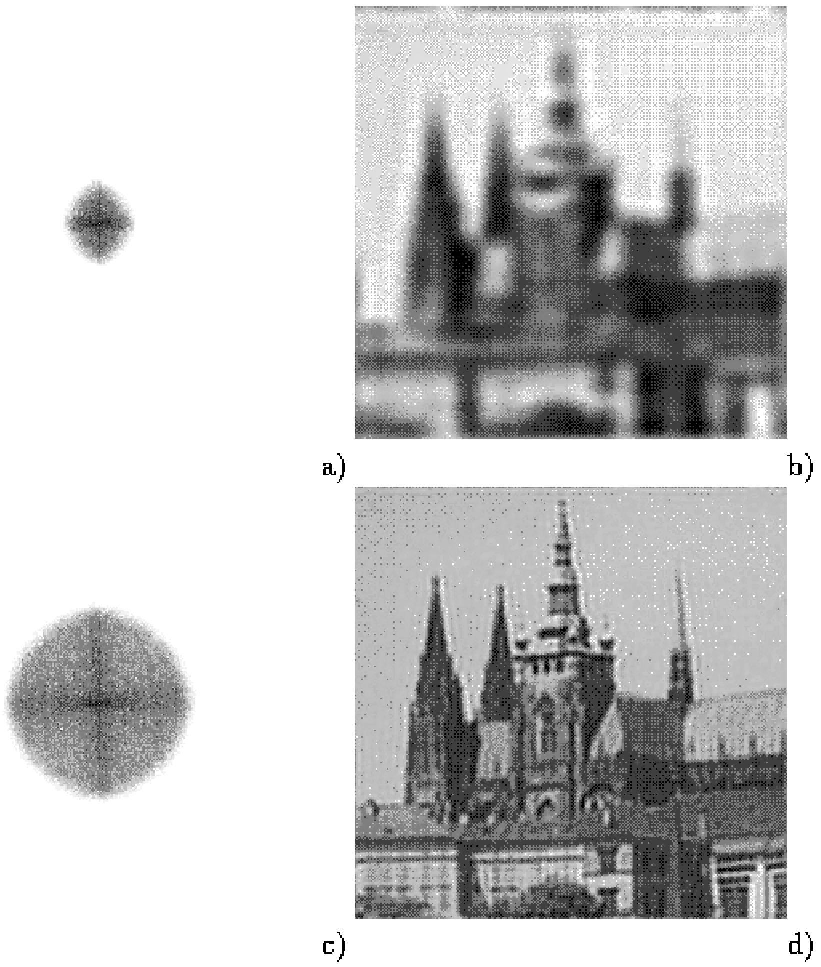


Figure 75: ILPF

**Problems**

- can not be implemented in electronic hardware
- cutting the frequencies very sharply results in artefacts (*ringing*)

  The shape of $h(x, y)$ (which determines the rings when convolved with a bright spot) depends on the value of $D_0$. The radii of the resulting rings are invers proportional to the value of $D_0$ (i.e.: small $D_0$ generates a low number of broad rings *strong ringing*). With increasing $D_0$ the number of rings increases but their breadth decreases.

**Butterworth Filter** (BLPF)

$$\hat{h}(u, v) = \frac{1}{1 + (D(u, v)/D_0)^{2n}}$$

The Buttermorth Lowpass Filter is a transferfunction of order $n$. There is no discontinuity and thus, less artefacts occur.

**Highpass Filter** By analogy to Lowpass filters, highpass filters allow high frequencies to pass, thus, edges and sharp transitions get emphasised.

$\hat{h}(u, v)$ is the *Ideal Highpass Filter* (IHPF) (see Fig. 76).

$$\hat{h}(u, v) = \begin{cases} 0 & D(u, v) < D_0 \\ 1 & D(u, v) \geq D_0 \end{cases}$$

**Butterworth Filter** (BHPF)

$$\hat{h}(u, v) = \frac{1}{1 + (D_0/D(u, v))^{2n}}$$

The Butterworth highpass filter ist a transfer function of order $n$. Edges and sharp transitions are kept and less artefacts occur. In order to retain a certain amount of lower frequencies, a constant value can be added to the transfer function (*High Frequency Emphasis*). Additionally, histogram equalisation can be applied to improve the result.

**Bandpass Filter** A specific (middle) frequency band is determined to pass and $\hat{h}(u, v)$ is designed correspodingly (fora result see Fig. 77).

More specific filtering techniques take specific properties of eventual disurbances into account (see e.g. Fig. 78).

### 3.5.3 Wavelet Transformation

**Motivation**

Using the Fourier transform, it is not possible to perform frequency filtering on a local scale. For that purpose, the **windowed Fourier transform** or short term Fourier transform (STFT) is used, which basically cuts the signal into pieces (by applying smooth window functions), and applies the Fourier

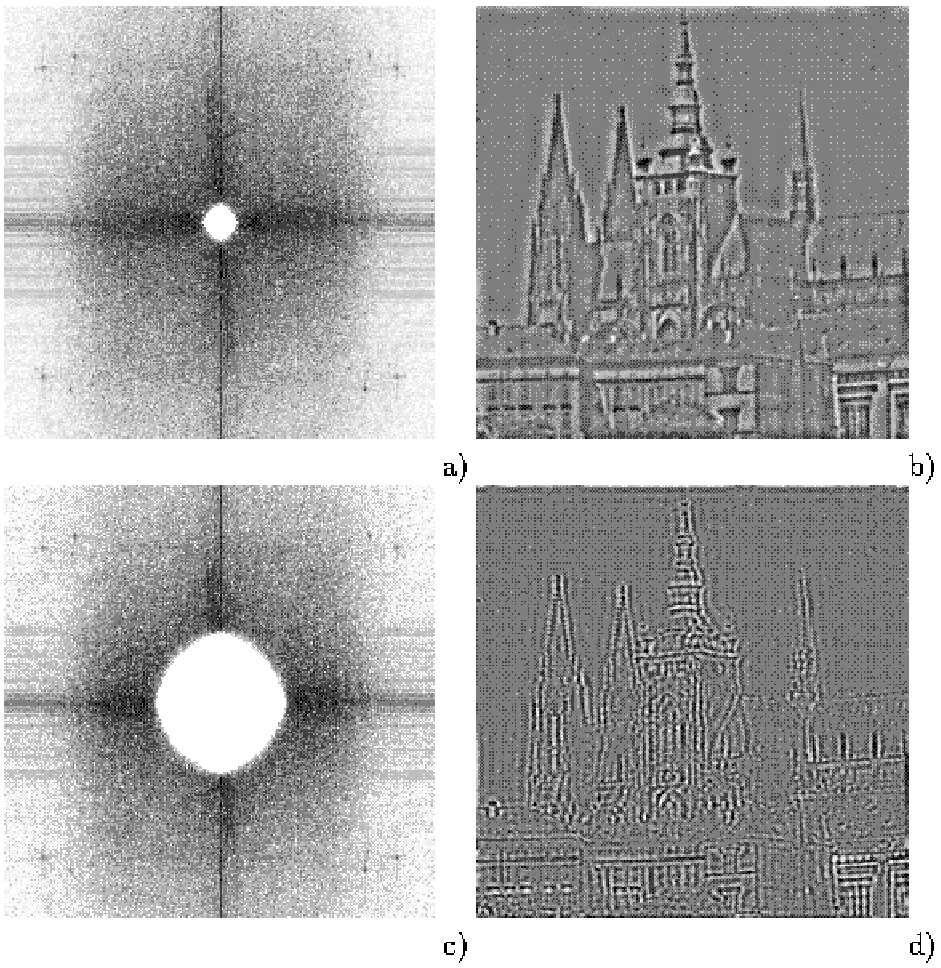


Figure 76: IHPF

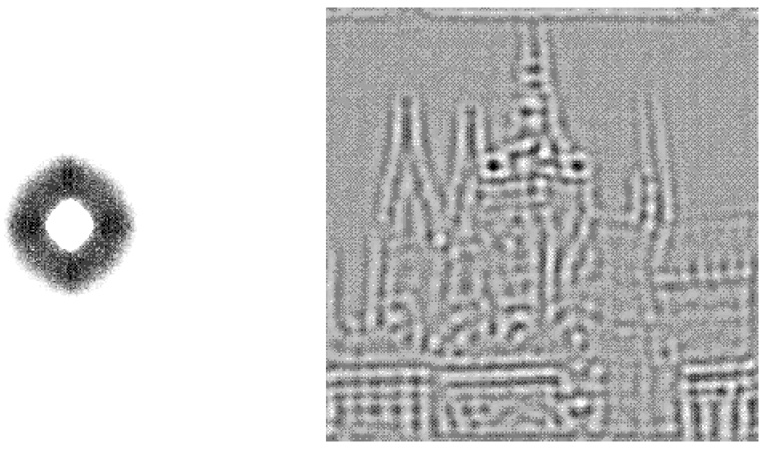

a) b)

Figure 77: BPF

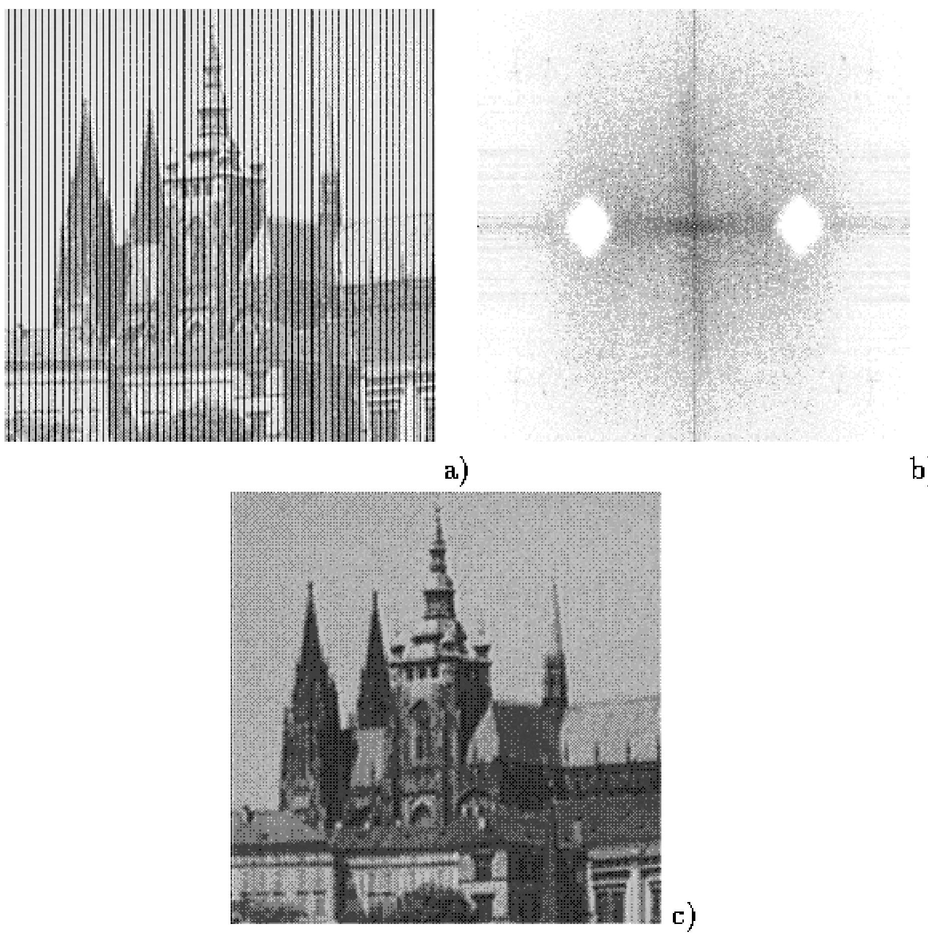

a) b) c)

Figure 78: Filtering specific frequency bands

Transform to the single peices. However, the width of the window functions have to be determined a priori, which results in frequency information loss no matter how the fixed window size have been chosen. A comprehensive solution to this dilemma is given by the Wavelet Transform.

$$W_{a,b}(f) = |a|_{-1/2} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-b}{a}\right) dt \tag{16}$$

$$\psi_{a,b}(s) = |a|_{-1/2}\psi\left(\frac{s-b}{a}\right) \tag{17}$$

The functions in equation (17) are named "Wavelets", $\psi(s)$ is the *mother wavelet*. Examples for mother wavelets are:

$$\psi(s) = (1-s^2)e^{\frac{s^2}{2}} \qquad \text{Mexican Hat} \tag{18}$$

$$\psi(s) = \frac{\sin(2\pi s) - \sin(\pi s)}{\pi s} \qquad \text{Shannon Wavelet} \tag{19}$$

$$\psi(s) = \begin{cases} 1 & 0 \le s \le 1/2 \\ -1 & 1/2 \le s \le 1 \\ 0 & \text{sonst} \end{cases} \qquad \text{Haar Wavelet} \tag{20}$$

The Wavelet transform depends on two parameters ($a$ and $b$). Changing $a$, the wavelets in equation (17) describe different local "frequency bands". Large $a$ describe broad, rather low frequent functions, small ones describe slim, fine detailed functions, rather representing high frequencies on a local scale. Changing parameter $b$ translates the time-space center of the function (which is in $s = b$). All wavelets are consequently translated and scaled versions of the mother wavelet.

**Multiresolution Analysis Idea**
Representation of signals as different levels of approximation (i.e. resolution) and the differences among thsoe resolutions. Orthogonal basis functions are being applied, parameters $a$ and $b$ are discretised: $a = a_0^m$ , $b = nb_0a_0^m$ with $m, n \in \mathbf{Z}$ and $a_0 > 1$, $b_0 > 1$. A common choice is $a_0 = 2$ and $b_0 = 1$.

$$W_{m,n}(f) = 2^{-m/2} \int_{-\infty}^{\infty} f(t)\psi(2^{-m}t - n)\, dt$$

A MRA is obtained by a series of approximation and detail spaces nested into each other, functions $\phi(t)$ and $\psi(t)$ are the respective orthonormal basis for these nested subspaces.

$$\phi(t) = \sum_n h(n)\phi(2t-n) \tag{21}$$

$$\psi(t) = \sum_n g(n)\phi(2t-n) \tag{22}$$

$g(n) = (-1)^n h(1-n)$
$\phi(t)$ ... scaling function
$\psi(t)$ ... wavelet function

The “Scaling Equation” relates the scaling function (dilated by factor 2) to its dilated and translated versions (functions with lower “frequency” are represented by identical functions with higher frequency). Important: The sequence $h(n)$ determines the resulting functions in a unique manner (i.e. given $h(n)$, $\phi(t)$ and $\psi(t)$ are uniquely defined).

**Fast Wavelet Transform** A fast wavelet transform in the one-dimensional case:

$$\begin{aligned} \text{input} &= (a, b, c, d, e, f, \ldots) \qquad\qquad h(n) = (1, 2, 3, 4) \\ WT_1 &= a + 2b + 3c + 4d \\ WT_2 &= b + 2c + 3d + 4e \end{aligned}$$

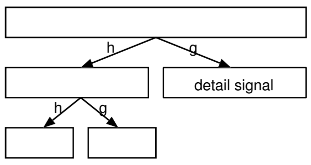


Figure 79: Wavelet Transform

Variants to handle border sample points in case of one-dimensional computation:

- Periodisation
- Signal extension
- Mirroring
- Zero-Padding

**2D Wavelet Transform** For images, a transform of a two-dimensional function is necessary. By analogy to the Fourier case (enabled by separable functions), the image is first transformed along the rows using a one-dimensional transform, subsequently the already transformed columns are transformed using again a one-dimensional transform. Usually, as it is the case for the one-dimensional transform, downsampling (with a factor 2) is applied. A visualisation of the entire process is shown in figure 80. An example for an image after wavelet transform is shown in figure 83.

**Filtering in the Wavelet Domain**

**Lowpass Filter** setting detail-subbands to 0.

**Highpass Filter** setting LL-subband (and low frequency detail-subbands) to 0.

**Bandpass Filter** Subband of interest for the application has to be retained, the rest is set to 0.

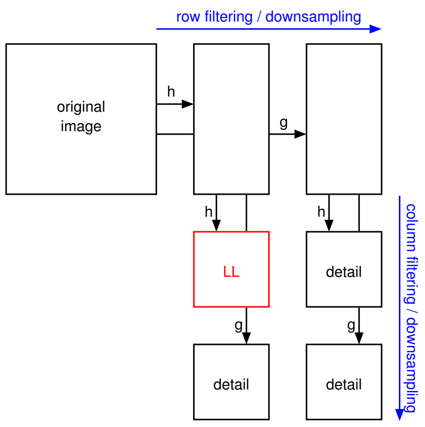


Figure 80: 2D Wavelet Transform

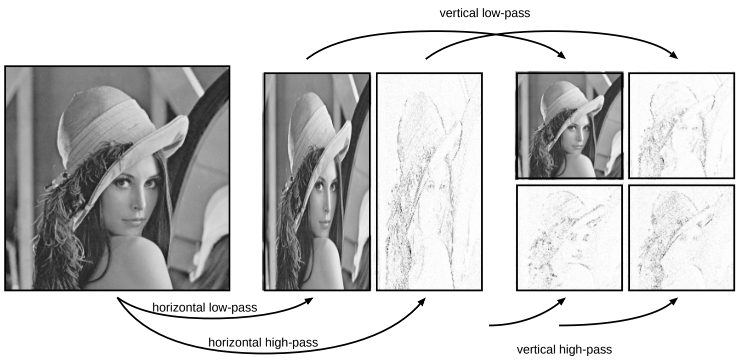


Figure 81: 2D Wavelet Transform: Visualisation 1. Level

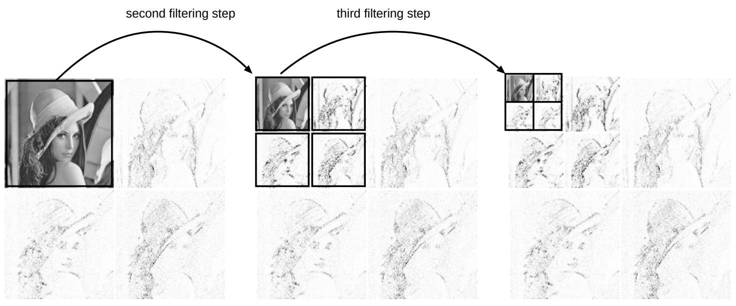


Figure 82: 2D Wavelet Transform: Visualisation 2.+3. Level

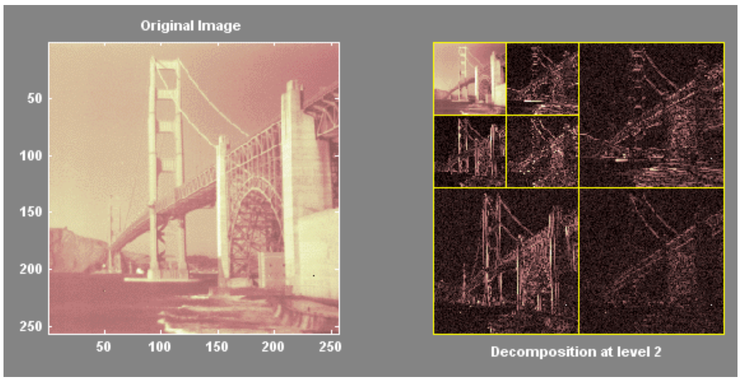


Figure 83: Wavelet Transform example

Remark: Setting the LL-Subband to 0, removes the graya-scale information entirely, only high frequency edge information is retained.

**Denoising** Denoising is achieved by thresholding in the wavelet domain. Only coefficients above a certain threshold are retained.

**Application** Wavelet Transform is used for signal decorrelation in in the context of compression in

- JPEG2000 and
- MPEG-4 VTC (visual texture coding).

Furthermore, wavelets are well suited for many tasks in signal- and image analysis.

### 3.5.4 Fourier vs. Wavelet

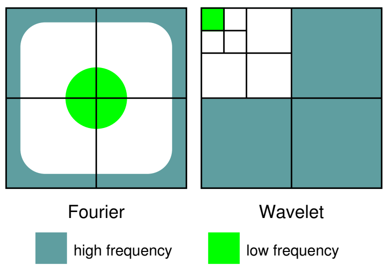


Figure 84: Fourier and Wavelet Transform

In case of the Fourier transform, a coefficient represents the global frequency content of the entire image with frequencies $u$ and $v$. In case of the wavelet transform, a coefficient represents the local frequency content at scale $2^i$ in a certain neighbourhood in the image. For this reason, in case an entire frequency band needs to be processed, Fourier methods are more appropriate, for local phenomena wavelet transforms are a better choice. In figure 84 the arrangement of the different frequencies for both transform domains is visualised.

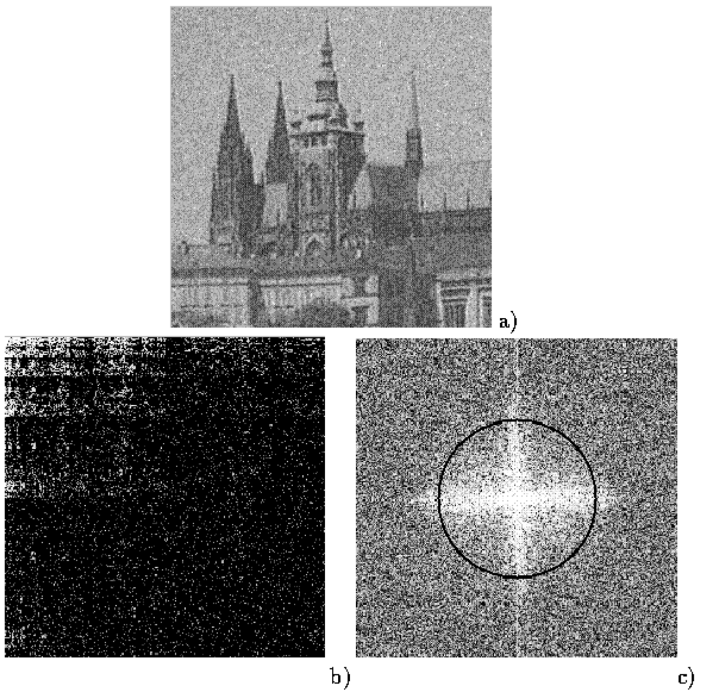


Figure 85: Denoising in the Wavelet/Fourier domain

### 3.5.5 Further Wavelet Transform variants

**Wavelet Packet Transform (WP)** Basic Idea: The iteration of the decomposition is not restricted to the low-pass subband but is applied to all wavelet subbands recursively. This leads to a much better (frequency) resolution, especially of the high frequency image parts.

Best Basis selection: Is a technique to choose the specific subtree for representing the signal, which allows to represent the signal in the most compact manner. iApplication is in compression techniques (FBI-standard, J2K Part II), subtrees are selected by optimising (information) cost functions.

Local Distcriminant Bases: Is a technique to choose the specific subtree for representing the signal, which allows to represent the signal in a way that allows to distinguish among different signal classes. The idea is to select the most discriminative features for a classification problem, texture classification is the most effective application area.

**A trous Algorithmus** The classical fast wavelet transform suffers from shift variance due to the downsampling stage in the transform. Shift invariance is

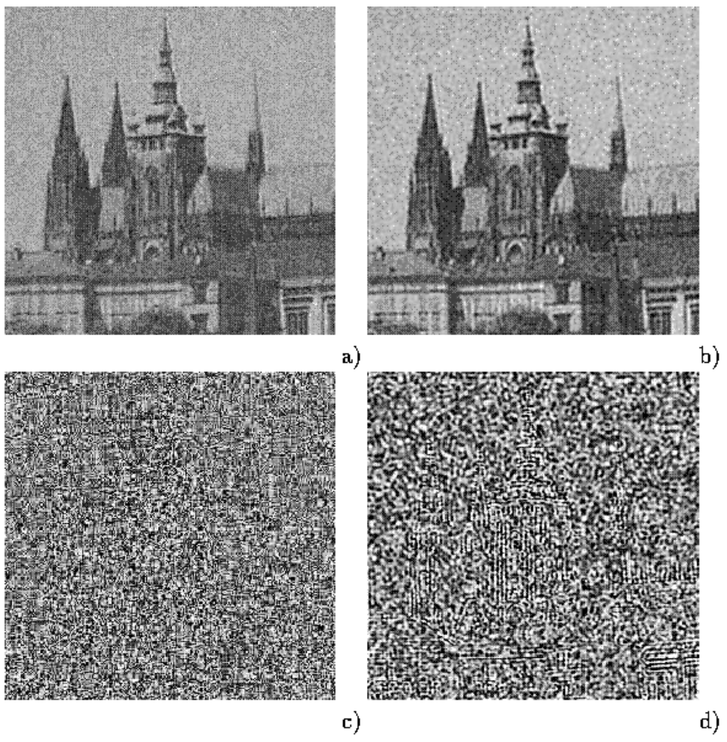


Figure 86: Denoising in Wavelet/Fourier domain: Results

achieved by omitting downsampling, by sacrificing the compact and redundant-free representation (in this manner, each decomposition level produces data of the same amount as the original signal), see Fig. 87. Contrasting to the CWT the fast wavelet transform (DWT) can be used. Scaling is coarse (powers of two, “octaves”).

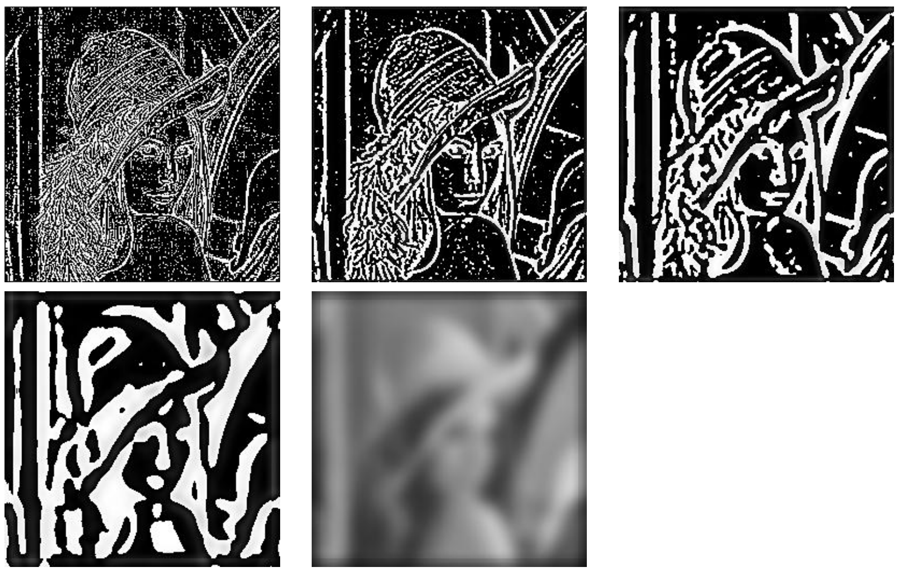


Figure 87: A trous Visualisation

**Continuous Wavelet Transform (CWT)** Direct computation of the transform coefficients with $O(N^2)$ complexity. For this algorithm, an actual wavelet in explicit formulation is required to compute the inner products between signal and basis function (note the difference to the DWT algorithm !). CWT is usually only used in one-dimensional application due to the large quantities of data produced and the high computational complexity.

# 4 Image Restauration

Image restauration improves image quality in case of an existing image distortion which should be removed. Contrasting to image enhancement the aim is to restore a (virtual) original image. Either the type of distortion is known or it has to be estimated. Reasons for existing distortions include: defocus, motion blur, noise (transmission errors, sensor noise, ...), defects in the optical system (Hubble), etc.).

**deterministic methods** for images with low amount of noise and known distortion function

**stochastic methods** try to identify the best restauration with respect to some statistical error-criterion (e.g. least-squares criterion)

The better the distortion is known, the better it can be removed and the better is the resulting restauration. In many cases, the distortion needs to be estimated:

**a priori estimation** distortion is known or is obtained before the restauration starts

**a posteriori estimation** image analysis based on *interesting* pixels (e.g. edges, straight lines, homogeneous areas) and it is attempted to estimate / reconstruct their original properties.

## 4.1 Image Distortion

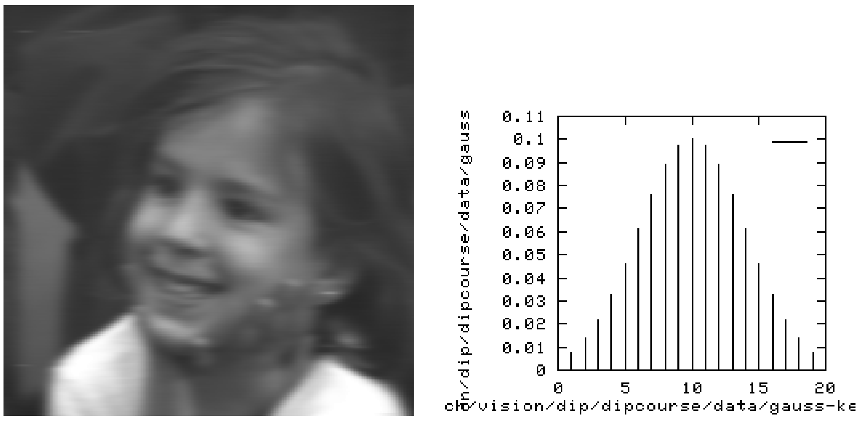
In the following scheme we assume a position invariant linear distortion and independent additive noise:

$$g(x,y) = h(x,y) * f(x,y) + v(x,y) \tag{23}$$

| | |
|---|---|
| $v(x,y)$ | ... noise |
| $h$ | ... distortion (position invariant) |

Following from linearity and the convolution theorem, we can represent the distorted signal in the DFT domain as follows:

$$\hat{g}(u,v) = \hat{h}(u,v) \cdot \hat{f}(u,v) + \hat{v}(u,v)$$



smoothed image Gauss kernel (1-D)

Figure 88: Example: Smoothing as image distortion

## 4.2 Distortion determination

There are several alternatives how to determine (estimate, approximate) the image distortion present. Since all these techniques are are approximative by nature, these ideas are called "blind deconvolution".

### 4.2.1 Image Analysis

In the distorted image we choose image regions with "obvious" image content, e.g. a sharp edge. The corresponding image part is denoted $g_s(x, y)$. For this image region we create an approximation $f_s^a(x, y)$ of the original image. Due to the selection of the region noise should not affect the result too much, thus we are able to compute the distortion function in the DFT domain for the region selected:

$$\hat{h}_s(u, v) = \frac{\hat{g}_s(u, v)}{\hat{f}_s^a(u, v)}$$

Due to the assumed position invariance, the distortion function can be generalised to the entire image.

### 4.2.2 Experimental distortion determination

Here we assume that the equipment used to take the distorted image is available (or at least the same or similar model). We capture an image similar to the one subject to restauration and try to generate a distortion highly similar to the one to model by systematic testing of the system configurations (e.g. different camera settings). Once identified, we take an image of an intensive point light source, to obtain the impulse response of the distortion (a distortion of the considered type is uniquely characterised by its impulse response). The DFT of an impulse is a constant A, thus we get:

$$\hat{h}(u, v) = \frac{\hat{g}(u, v)}{A}$$

### 4.2.3 Distortion determination by modelling

Here we exploit knowledge about models of physical processes.

Examples for simple distortions are ...

**relative (uniform) motion between camera and object** $f(x,y)$ moves in a way such that $x_0(t)$ and $y_0(t)$ are the time-dependent motion components in x and y direction.

The entire image exposition is obtained by intergating the image function over the time-frame of the shutter being opened $T$:

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t))dt$$

$$\hat{g}(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-2\pi i(ux+vy)}dxdy$$

$$\hat{g}(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[\int_0^T f(x - x_0(t), y - y_0(t))dt\right] e^{-2\pi i(ux+vy)}dxdy$$

The integration order can be swapped:

$$\hat{g}(u,v) = \int_0^T \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x - x_0(t), y - y_0(t))e^{-2\pi i(ux+vy)}dxdy\right] dt$$

The expression between [] is the DFT of the shifted function $f(x-x_0(t), y-y_0(t))$. Using the translation properties of the DFT and the independence between $\hat{f}(u,v)$ and $t$ we obtain

$$\hat{g}(u,v) = \int_0^T \hat{f}(u,v)e^{-2\pi i(ux_0(t)+vy_0(t))}dt = \hat{f}(u,v)\int_0^T e^{-2\pi i(ux_0(t)+vy_0(t))}dt$$

Thus, we set $\hat{h}(u,v) = \int_0^T e^{-2\pi i(ux_0(t)+vy_0(t))}dt$. Setting for example $x_0(t) = at/T$ and $y_0(t) = 0$ we get motion only in x-direction (Ex.: Taking pictures from a moving car). At time $t = T$ the image moved by distance $a$. We get

$$\hat{h}(u,v) = \int_0^T e^{-2\pi iuat/T} = \frac{T}{\pi ua}\sin(\pi ua)e^{-\pi iua}$$

For two-dimensional motion (also $y_0(t) = bt/T$ as well) we get:

$$\hat{h}(u,v) = \frac{T}{\pi(ua+vb)}\sin(\pi(ua+vb))e^{-\pi i(ua+vb)}$$

**Defocus**

$$\hat{h}(u,v) = \frac{J_1(ar)}{ar} \text{ mit } r^2 = u^2 + v^2$$

$$J_1(x) = \sum_{k=0}^{\infty}\frac{(-1)^k(x/2)^{2k+1}}{k!(k+1)!}$$

$J_1$ ... Bessel function Order 1
$a$ ... Extent of Defocus

**Atmospherical Turbulence**

$$\hat{h}(u,v) = e^{-c(u^2+v^2)^{5/6}}$$

## 4.3 Distortion Removal

In order to remove the distortion, we need to construct a restoration filter, exhibiting a transfer function invers to the distortion: $\hat{h}^{-1}(u,v)$. The resulting procedure is denoted "Inverse Filtering".

$$\hat{f}(u,v) = \hat{g}(u,v) \cdot \hat{h}^{-1}(u,v) - \hat{v}(u,v) \cdot \hat{h}^{-1}(u,v)$$

In case the noise contribution is not too high, restauration is identical to inverse convolution.

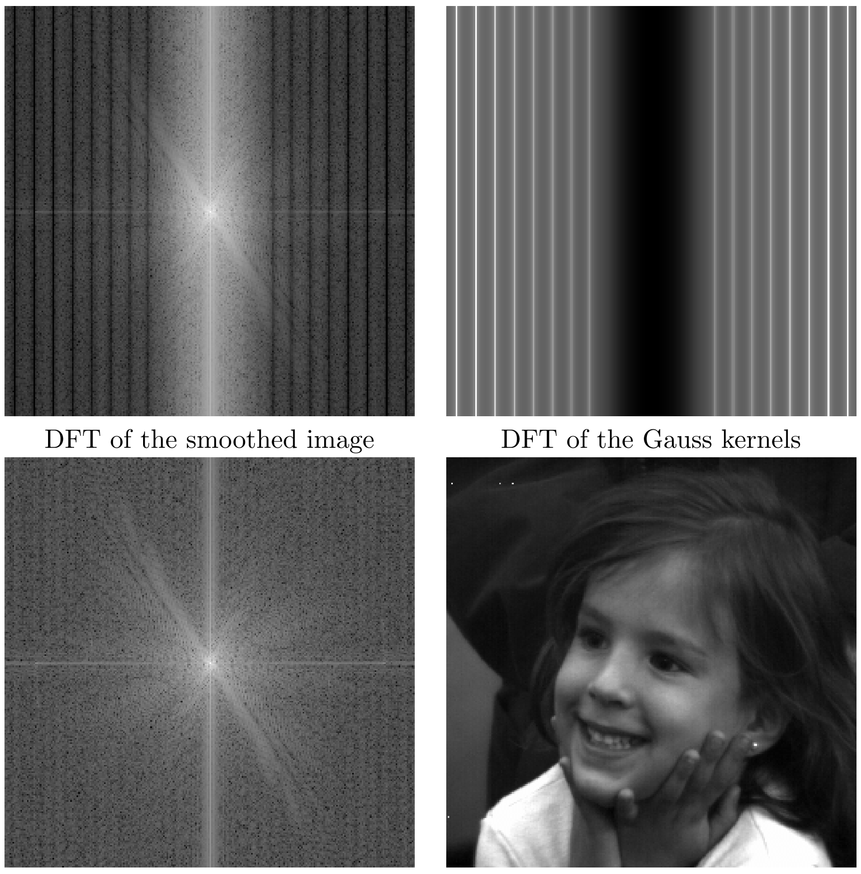


Figure 89: Beispiel Inverse Filterung

In case of the noise contribution is too high or $\hat{h}(u,v)$ is too small we result in a large value for $\hat{v}(u,v) \cdot \hat{h}^{-1}(u,v)$ which dominates the inverse filtering. Fig. 90 shows the result of inverse filtering with high noise contribution: the smoothed image (originally given as (`float` data type) has been casted to `char` data type, resulting in significant (quantisation) noise. The result of inverse filtering is quite poor as a result.

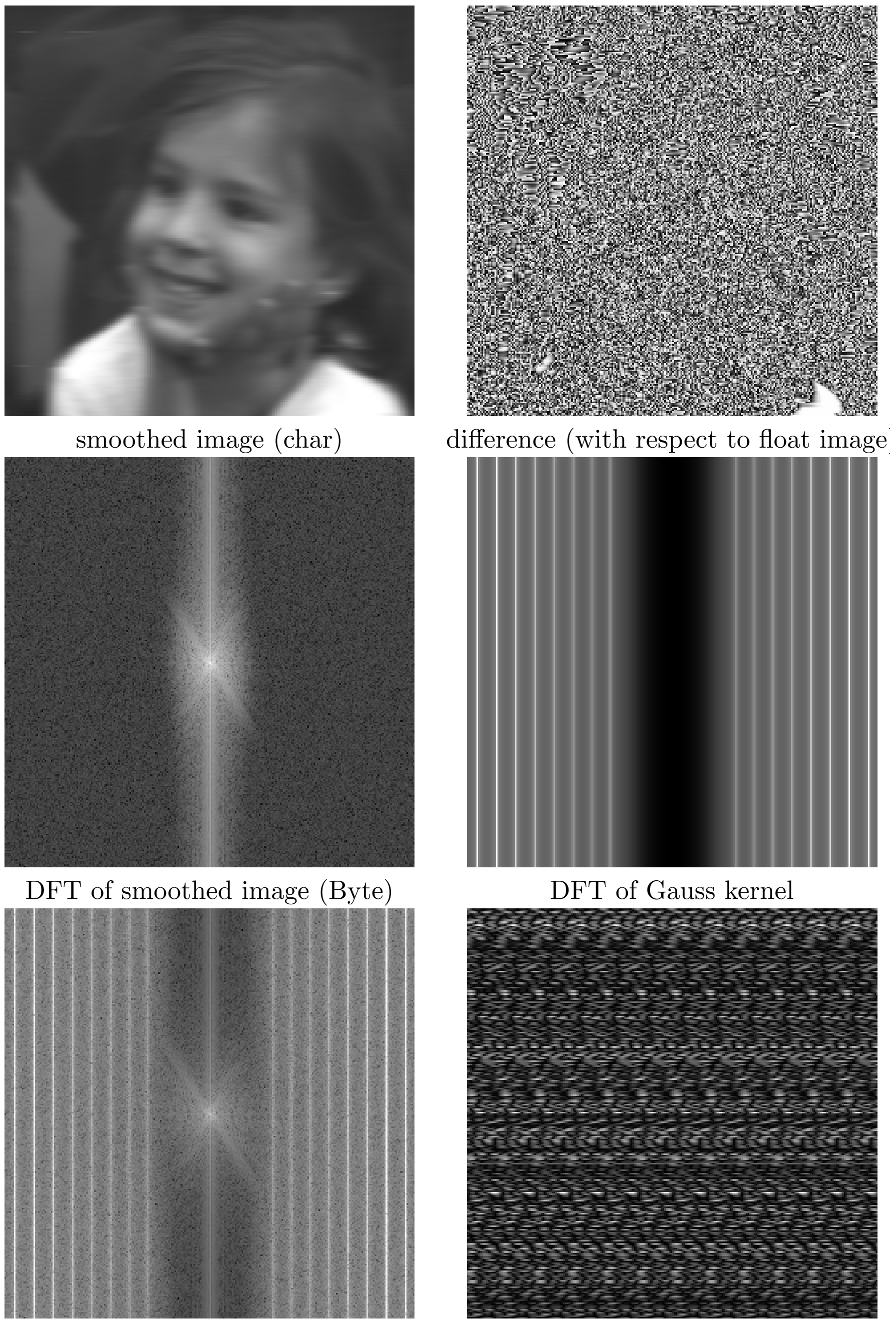

smoothed image (char)
difference (with respect to float image)
DFT of smoothed image (Byte)
DFT of Gauss kernel
DFT after inverse filtering
image after restauration

Figure 90: Example for Inverse Filtering with noise

This phenomenon leads to the definition of the "Pseudo-inverse Filtering":

$$\hat{h}^{-1}(u,v) = \begin{cases} h^{-1}(u,v) \ if \ |\hat{h}(u,v)| > T \\ 0 \ if \ |\hat{h}(u,v)| \leq T \end{cases}$$

Obviously, the case of $\hat{h}(u,v)$ being too small is "corrected" to prevent a large value for $\hat{v}(u,v) \cdot \hat{h}^{-1}(u,v)$. Still, large values for $\hat{v}(u,v)$ still remain unsolved (noise contribution),

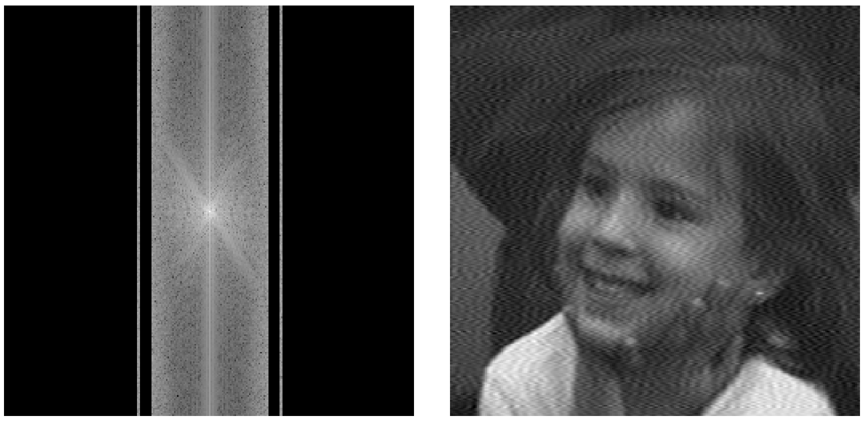

DFT after pseudo-inverse Filtering Image after pseudo-inverse Filtering

Figure 91: Example: Pseudo-Inverse Filtering

## 4.4 Wiener Filtering

Wiener filering additionally exploits a priori knowledge about the noise contribution. This type of restauration delivers an estimation of the non-distorted image $\bar{f}$ with minimal error $f(i,j) - \bar{f}(i,j)$ with respect to some determined metric. $s_{xx}$ and $s_{\eta\eta}$ are the spectral densities of the noise and the original (non-distorted) images. Of course, these values are difficult to obtain.

$$\hat{\bar{f}}(u,v) = \hat{h}_w(u,v) \cdot \hat{g}(u,v) \tag{24}$$

$$\hat{h}_w(u,v) = \frac{\hat{h}^*(u,v)}{|\hat{h}(u,v)|^2 + \frac{s_{xx}(u,v)}{s_{\eta\eta}(u,v)}} \tag{25}$$

In order to be able to conduct this type of filtering, information about the distortion and statistical knowledge about the noise are required. An example is given in figure 92.

Since the actual knowledge required for this process (e.g. computation of the spectral density of the noise) might be hard or impossible to obtain, a "parameterised" Wiener filter can be employed:

$$\hat{h}_w^K(u,v) = \frac{\hat{h}^*(u,v)}{|\hat{h}(u,v)|^2 + K}$$

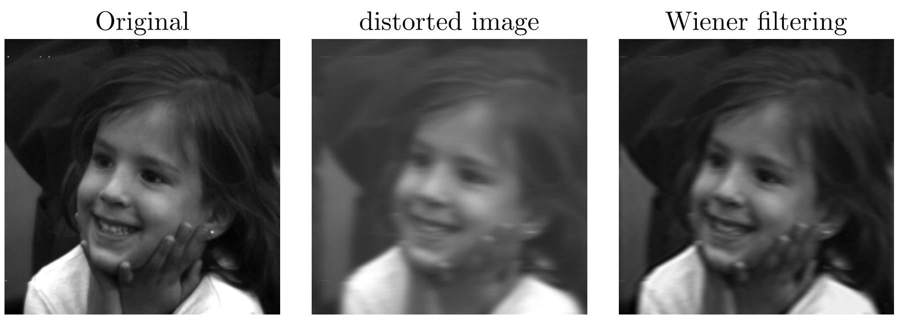


Figure 92: Wiener Filtering

In the process, K is optimised until the best result is reached. This can be improved by using so-called “constrained least square” filters. Intuitively this means that the optimisation of $K$ is done with respect to some least squares criterium, e.g. to maximise image smoothness (or minimisation of image variations), expressed in terms of a gradient operator.

# 5 Edge Detection

The terms *edge* and *crack edge* have been already introduced in Chapter 2.3.3.

To recall that: Edges are pixels, in which the image intensity function changes its magnitude. Crack edges are a virtual edge entity between pixels (see figure 47).

The are three different types of gradient operators:

1. Operators approximating the derivative of the image intensity function by differences: $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
2. Operators approximating the zero-crossings of the second derivative of the image intensity function: $\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$
3. Operators mapping the image intesity function to a paramertised edge model

## 5.1 Techniques using the 1. derivative

### 5.1.1 Roberts Operator

The Roberts operator (see figure 95) uses a $2 \times 2$ neighbourhood with two convolution masks thereby not considering the orientation of the edges.

**Disadvantage**
High sensitivity against noise, since only a low number of pixels is used in the approximation.

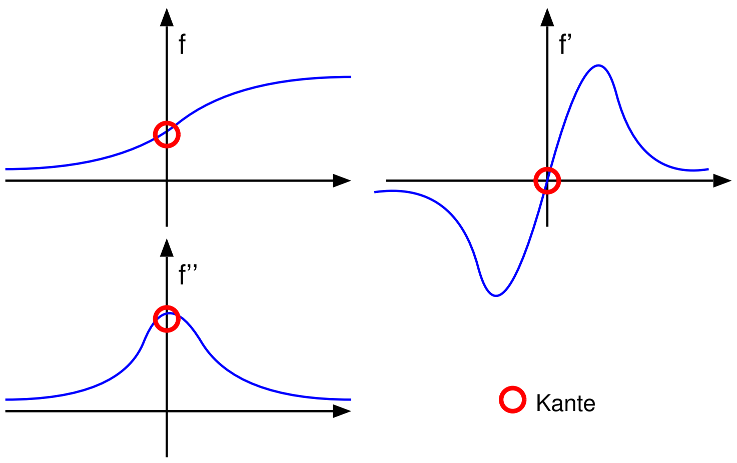


Figure 93: Visuell: 1. derivative vs. 2. derivative (try to identify two errors in this graphic !)

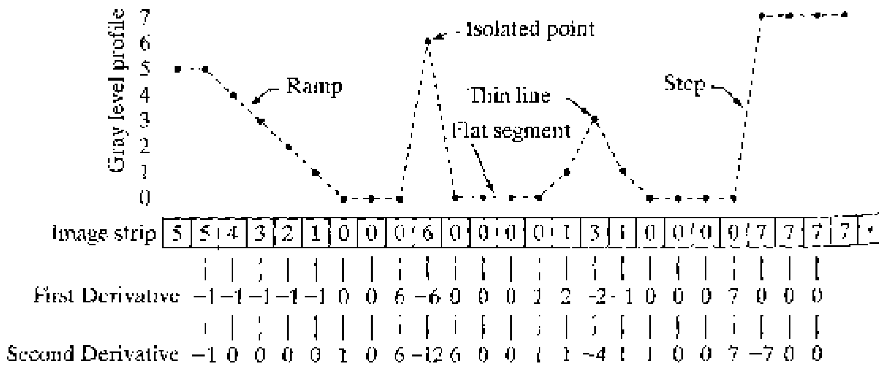


Figure 94: Numerics: 1. vs. 2. derivative

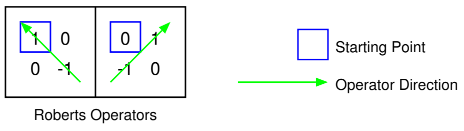


Figure 95: Roberts Operator

### 5.1.2 Compass operators

The following operators are called *Compass Operators*, since they determine the orientation of the gradient. A mask is applied in eight orientations, the largest response determines the gradients orientation.

**Prewitt operator** (see figure 96)

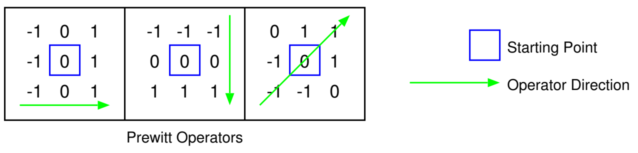


Figure 96: Prewitt operator

**Sobel operator** (see figure 97 and figure 98)

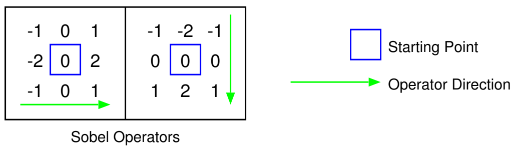


Figure 97: Sobel operator

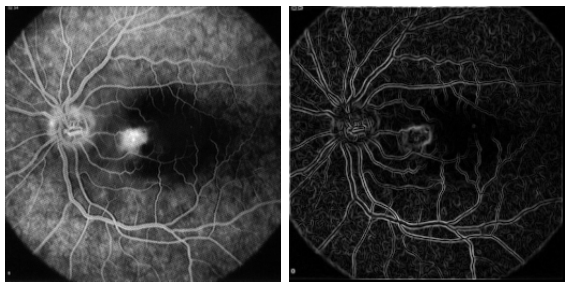


Figure 98: Sobel operator example

**Robinson operator**

$$h_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix}$$

**Kirsch operator**

$$h_1 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{pmatrix}$$

In figure 99 we visualise the application of various edge detectors.

**Disadvantages**

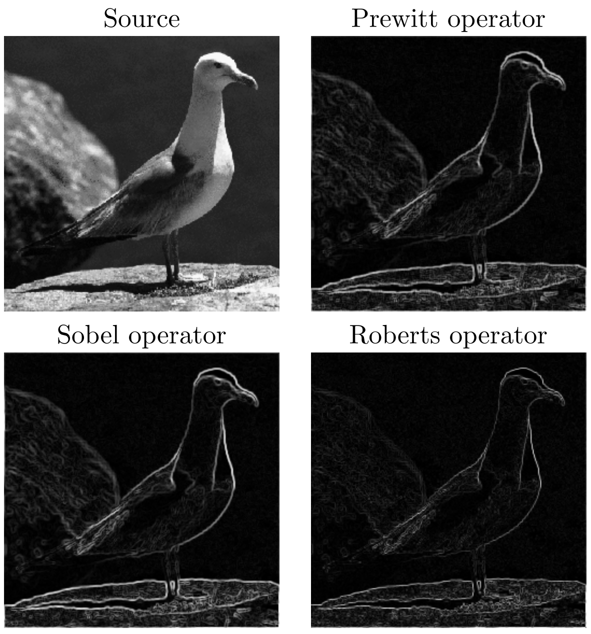


Figure 99: Edge detection examples

- Sensitivity against noise
- Sensitivity againest the size of the object and scale / type of the edge (step edge vs. ramp edge). In many cases it is easier to locate zero-crossings than maxima or minima (see figure 94).

## 5.2 Techniques using the 2. derivative

### 5.2.1 Laplace Operator

If an application only requires the magnitude of the gradient regardless of its orientation, the Laplace operator is a good choice, which is rotation invariant (recall: it can be computed effectively in the Fourier domain).

$$\bigtriangledown^2 (x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{26}$$

In most cases, it is approaximated using 3×3 masks for 4- und 8-neighbourhoods:

$$h_4 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad h_8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

### 5.2.2 Mexican Hat Operator

The Marr-Hildreth Operator (also denoted as Mexican Hat Operator) uses a two-dimensional Gauss filter as a smoothing operaor:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The standard deviation $\sigma$ is proportional to the size of the neighbourhood the filter is operating on.

To compute the second derivative (in order to identify zero-crossings) the Laplace operator is applied to the smoothed image:

$$\bigtriangledown^2 \left(G(x, y, \sigma) * f(x, y)\right) \overset{\text{linear}}{\rightarrow} \left(\bigtriangledown^2 G(x, y, \sigma)\right) * f(x, y)$$

$\bigtriangledown^2 G$ is image independent and thus can be computed in advance.

$$\begin{aligned} r^2 &= x^2 + y^2 \\ G(r) &= e^{-\frac{r^2}{2\sigma^2}} \\ G'(r) &= -\frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \\ G''(r) &= \frac{1}{\sigma^2}\left(\frac{r^2}{\sigma^2} - 1\right) e^{-\frac{r^2}{2\sigma^2}} \end{aligned}$$

replacing $r^2$ by $x^2+y^2$ again, we obtain the *Laplacian of Gaussian* (LoG), which resambles the shape of a sombrero, i.e. Mexican Hat (see figure 100).

$$h(x, y) = \frac{1}{\sigma^4}\left(\frac{x^2+y^2}{\sigma^2} - 1\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{27}$$

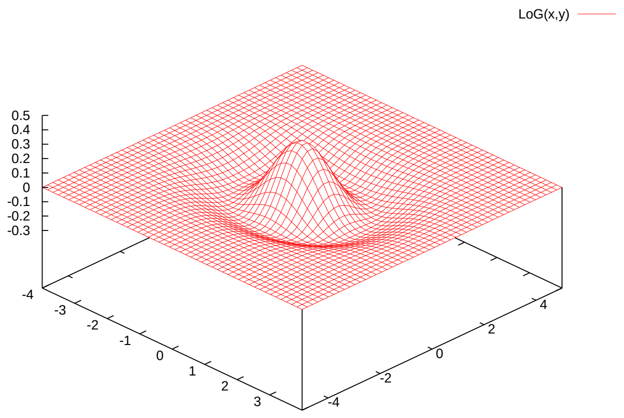


Figure 100: Mexican Hat

Fig. 101 shows some examples for increasing values of $\sigma$, the larger those values, the more responses we get from coarse edges (large $\sigma$ in the Gauss mask leads to a strong smoothing effect).

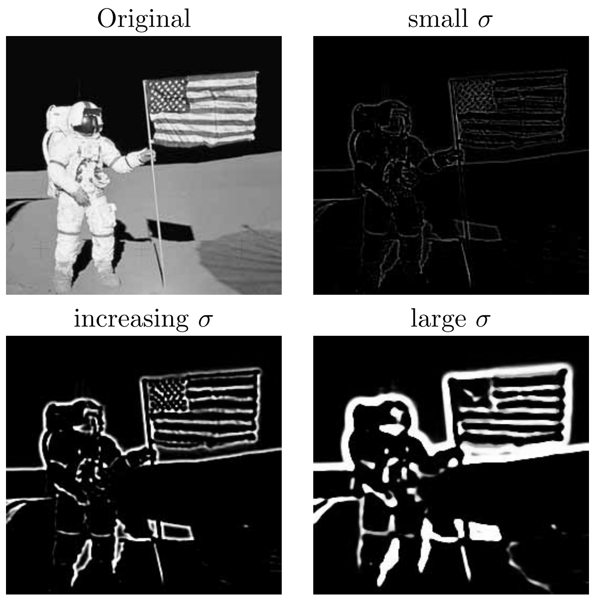


Figure 101: LoG Example

Remark: $\bigtriangledown^2 G$ can be approximated efficiently by a difference of two Gauss-masks with different *sigma* (*Difference of Gaussian*).

**Advantage**
By selecting $\sigma$, it is possible to set the scale with respect to which the edge property should be determined.

**Disadvantage**

- significant smoothing
- trend to form closed curves (*plate of sphagetti*)

## 5.3 Canny Edge Detector

The Canny Edge Detector (see figure 103 for an example) has been developed in 1986 and has been theoretically proven to be optimal with respect to the following criteria for noisy *step edges*

**Detection** no important edges are missed and no false edges are listed (low value for false negative and false positive edges)

**Localisation** distance among actual edge position and computed edge position is minimal

**One response** multiple responses to one edge are minimised

Canny Edge Detector achieves these results by applying the following techniques:

**thresholding with hysteresis** improves detection performance in the presence of noise. Edge pixels have to satisfy the following conditions:

- edge-magnitude > *high threshold*
- edge-magnitude > *low threshold* and is connected to an edge pixel > *high threshold*

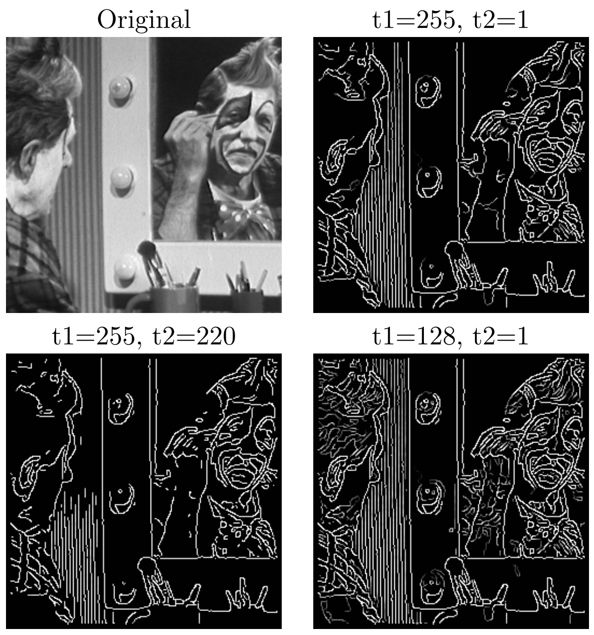


Figure 102: Edge Detection Examples: thresholding

**non-maximal suppression** only local maxima of the edge image are processed further.

**feature synthesis approach** all edges with respect to a small scale (i.e. fine detail edges, small $\sigma$) are marked. A predictor for larger $\sigma$ (see section 5.2.2) is used to predict edge pixels for a larger $\sigma$: Smoothing of the small scale edges. In the comparison to the actual computed values only those values are additionally kept as compared to the predicted ones, which are significantly larger. This procedure is conducted for several values of $\sigma$.

**Algorithm**

1. iterate 2 until 5 for increasing values of $\sigma$
2. convolve the image with Gaussian with $\sigma$
3. compute gradient magnitude and direction
4. find position of the edges (non-maximal suppression)
5. thresholding with hysteresis
6. feature synthesis approach

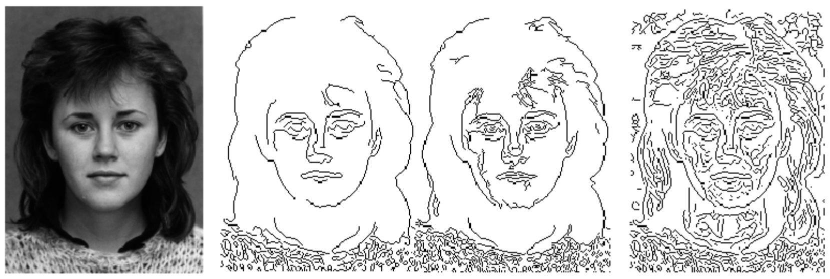


Figure 103: Canny Edge Detector: different $\sigma$

For a demo: `http://www.ii.metu.edu.tr/~ion528/demo/lectures/6/4/` and `http://www.cs.washington.edu/research/imagedatabase/demo/edge/`.

## 5.4 Line Finding Alogrithms

### 5.4.1 Simple Kernels

A **line** is a curve that does not bend sharply.

In case a line has a width of one or two pixels, a gradient image can be generated using the following function and mask:

$$f(i,j) = \max(0, \max_k (g * h_k)$$

$$h_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

### 5.4.2 Hough Transformation

Hough transform maps edge pixels to a parametric model of a curve of a certain type. In order to be applied, we have to know which curves we are looking for.

A **straight line** is defined by two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$. All lines passing through $A$ are given by $y_1 = mx_1 + c$, for arbitrary $m$ and $c$ (this equation is associated with the parameter space $m, c$). In this representation, all lines through $A$ are given by $c = -x_1m + y_1$, those through $B$ are given by $c = -x_2m+y_2$. The only common point of those two lines in the $m, c$ parameter space is the point representing the line connecting $A$ and $B$. Each line the the image is represented by a single point in the $m, c$ parameter space.

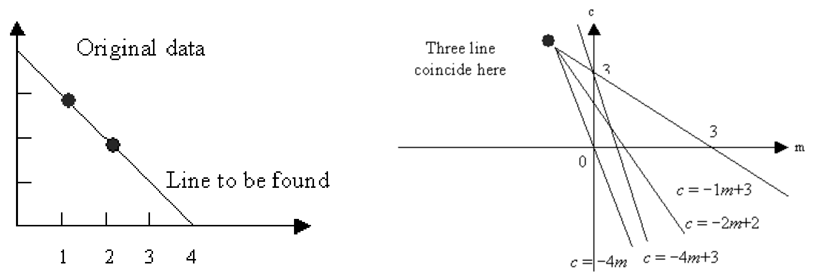


Figure 104: Fundamental idea of the Hough transformation

The points (1,3), (2,2) and (4,0) are situated on the line we are looking for, while (4,3) is not.

| y | x | Gives | Transposed |
|---|---|---|---|
| 3 | 1 | 3 = m 1 + c | c = -1 m + 3 |
| 2 | 2 | 2 = m 2 + c | c = -2 m + 2 |
| 0 | 4 | 0 = m 4 + c | c = -4 m |
| 3 | 4 | 3 = m 4 + c | c = -4m + 3 |

Based on the line-definition as discussed so far, we obtain the following procedure:

1. determine all potential line pixels (edge detection)
2. determine all lines passing through these line pixels
3. transform these lines into $(m, c)$-parameter space
4. determine point $(a, b)$ in parameter space, which is the result of the Hough transform of the line $y = ax+b$ ($(a, b)$ is the only tupel that occured several times).

Remark: Only a limited number of lines is considerd passing through the line pixels. Collecting the parameters of all admissible lines results in an *accumulator array*, the elements of which are denored *accumulator cells*.

For each line pixel, potential lines which point into an admissible direction are determined and the parameters $m$ and $c$ are recorded and the value of the corresponding accumulator cell $A(m, c)$ is incremented. Lines in the image are found by taking local maxima of the accumulator array.

**Advantage**
Robust against noise

**Advantage and disadvantage**
Missing line parts are interpolated

In actual iplementations the usage of the common line equation $y = mx + c$ für $m \to \infty$ is suboptimal in case of large slope. The line equation $r = x\cos(\theta) + y\sin(\theta)$ is better for such cases. Check out the visualisation:
`http://www.vision.ee.ethz.ch/~buc/brechbuehler/hough.html`

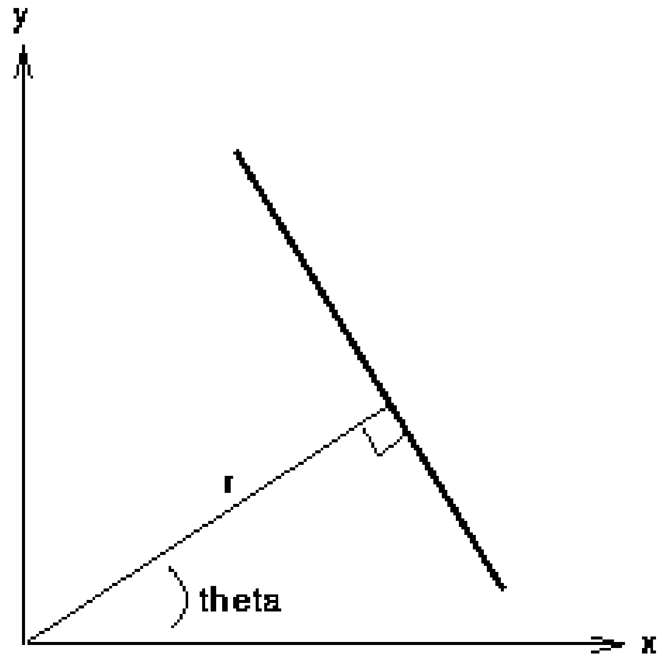


Figure 105: Alternative line formula

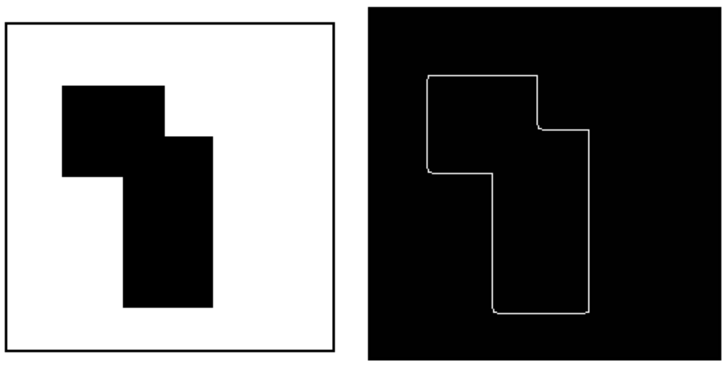


Figure 106: Example for Hough transform

In the general setting the curve equation $f(x, a) = 0$ with $a$ being the vector of curve parameters is used.

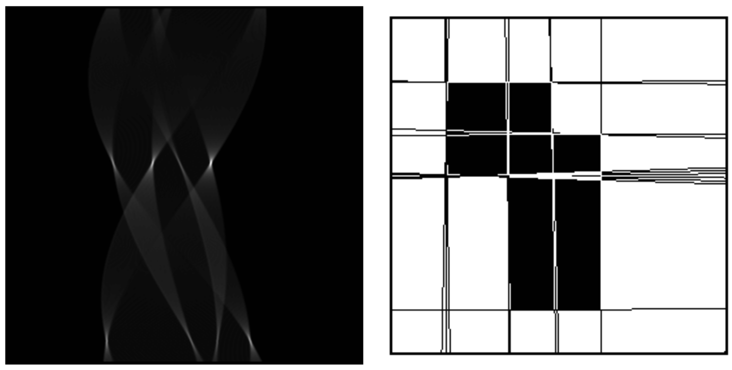


Figure 107: Example for Hough transform

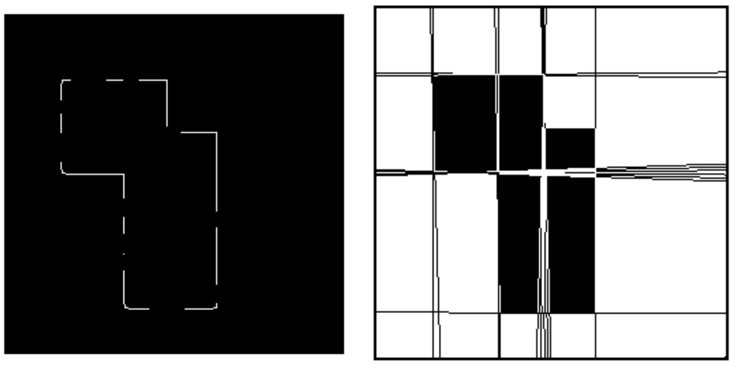


Figure 108: Example for Hough transform

**Algorithm**

1. Quantisation of the parameter space, dimension $n$ of this space is the number of parameters in $a$
2. generate the $n$-dimensional accumulator array $A(a)$ and set $A(a) = 0$
3. $\forall(x_1, y_1)$ in the appropriately thresholded gradient image increase accumulator cell $A(a)$ in case of $f(x, a) = 0$
4. local maxima in the accumulator array are the curves we have been looking for

For circles $(x_1 - a)^2 + (y_1 - b)^2 = r^2$ three parameters and a corresponding three dimensional accumulator array are required. By analogy, for more complicated curves, high dimensional accumulator cells are used. Due to the high complexity, often local variants of the Hough transform are used.

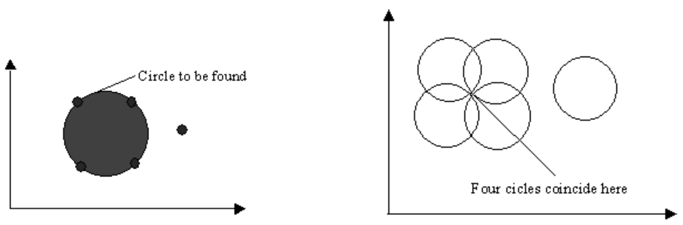


Figure 109: Circular Hough transform

# 6 Image Segmentation

Image segmentation is one of the most important stages in the image analysis process. The main aim is to identify parts of the image exhibiting a high correlation with real world objects or areas. The key to success is to grasp (at least parts of) the semantics of the image content.

**complete segmentation** a set of regions / objects with empty intersection which uniquely correspond to real objects. In many cases expert knowledge applied in artificial intelligence methods is required. However, there are some applications which are simple enough to generate good results with simpler approaches, e.g. text with letters and numbers, objects in front of uniform background and high contrast in general.

**partial segmentation** : regions which are homogeneous with respect to some criterion are identified, however, to do not necessariliy correspond to objects of the scene (in many cases, over-segmentation occurs, i.e. to many regions are found).

Segmentation techniques can be grouped into three classes:

1. Techniques which use global information about the image or parts of it (e.g. the histogram)

2. Edge-based techniques try to generate closed edge-chains as object borders
3. Region-based techniques try to generate homogeneous regions (the latter two techniques solve a dual problem: each region can be described by its closed border curve and each curve describes a region which is enclosedby the curve.

## 6.1 Thresholding

Thresholding is the simplest segmentation technique, but due to its high speed it is still of importance. An intensity constant or threshold is determined to separate objects and background. Thus, the input image is transformed into a binary segemented output image as follows:

$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases}$$

Thus, thresholding is a suited approach in case objects are not joined and object gray-scale is different from background gray-scale.

Of course, a central question is the selection of the threshold. Using a fixed threshold is successful in rare cases only. Variable thresholding is often more successful ($T = T(f, f_c)$ with $f_c$ for the image part considered) which varies the threshold value as a function of changing image part characteristic.

## 6.2 Thresholding Variations

**Band Thresholding** gray-scales in a specific gray-scale range / band are determined to be object pixels (as opposed to background or non-interest pixels). For example, segmentation of cells or cell parts in microscopic images, known gray-scales caused by known material density in CT-images)

$$g(i,j) = \begin{cases} 1 & f(i,j) \in D \\ 0 & \text{otherwise} \end{cases}$$

**Multi Thresholding** employs several thresholds and the output image is not binary.

$$g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \in D_1 \\ 2 & \text{for } f(i,j) \in D_2 \\ 3 & \text{for } f(i,j) \in D_3\text{etc.} \end{cases}$$

**Semi Thresholding** The aim is to remove background but to keep gray-scale information in the objects

$$g(i,j) = \begin{cases} f(i,j) \text{ for } f(i,j) \geq T \\ 0 & \text{otherwise} \end{cases}$$

## 6.3 Threshold Selection

In a text, it is known that letters cover a certain share of the text image pixels ($1/p$ of the image area). In this situation, it is straightforward to select the threshold by analysing the histogram, such that $1/p$ of the image is $\leq T$ (is denoted "p-tile thresholding").

However, in most cases we do not have such informations. Thus, threshold selection is done by analysing the histogram:

- Objects with uniform gray-scale which is different from the (uniform) grayscale of the background: the histogram is *bimodal*. The threshold is selected to be the minimal value between the two extrema.
- In case of *multimodal* histograms thresholds need / can be selected between two corresponding maxima – either, different segmentation results are obtained or multithresholding has to be applied.

**Problem**
How can we decide if a histogram is bimodal or multimodal ? Usually, bimodal techniques take the largest two maxima and set $T$ as a minimum in between ("mode method"). In order to avoid that two close local maxima are selected, a minimal distance in terms of gray-scales should be enforced. Alternatively, the histogram can be smoothed. Attention: a bimodal histogramm does not guarantee a correct segmentation, e.g. 50% B/W pixel mixed or concentrated on one image side result in an identical bimodal histogramm.

**Further Strategies to Select Thresholds**

**local neighbourhoods** Consideration of local neighbourhoods when generating the histogram (e.g. by assigning weights to pixels in order to supress pixels with large gradient - in this case the histogram does not contain pixels of the border curve but only object and background pixels)

**Optimal Thresholding** Histogram is approximated by a weighted sum of probability densities – the threshold is selected to be the minimal probability between the maxima of the distributions.

**Problem**: Estimation of the parameters of the distributions and identification of the type of the distribution (normal distribution is often assumed).

**Iterative Threshold Selection** We assume the existence of regions with two dominating gray-scales. The corners of the image contain background pixel.

In iteration $t$ we compute the averages $\mu_B^t$ (background) and $\mu_O^t$ (object),

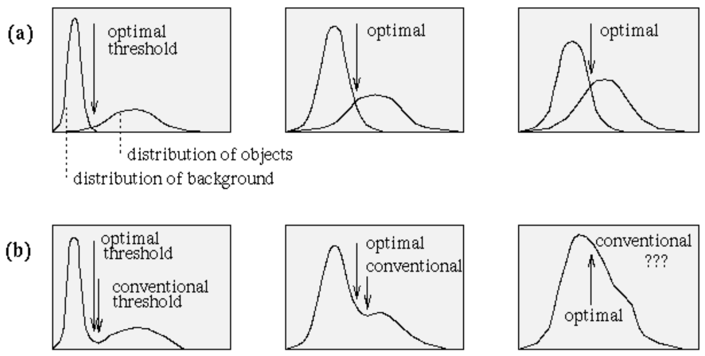


Figure 110: Concept of optimal thresholding

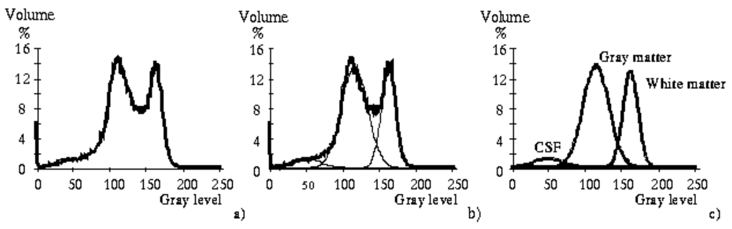


Figure 111: Example for optimal thresholding: Histogram

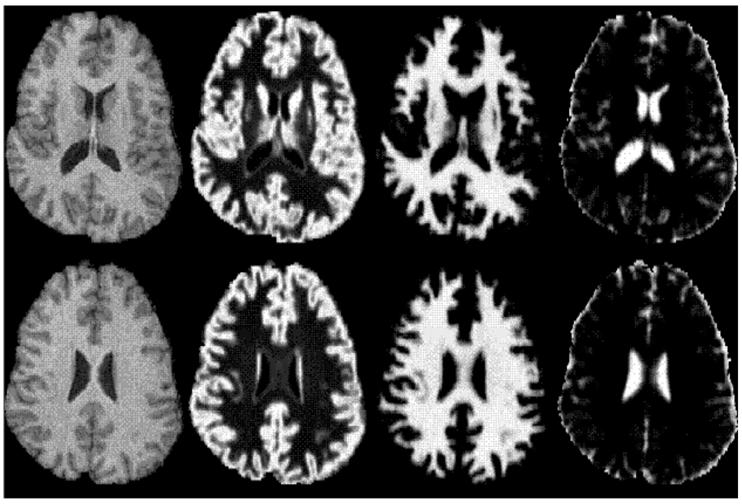


Figure 112: Example for optimal Thresholding: Result

and the segmentation in step $t$ is defined to be

$$\begin{aligned}
T^t &= \frac{\mu_B^{t-1} + \mu_O^{t-1}}{2} \\
\mu_B^t &= \frac{\sum_B f(i,j)}{\text{number of background pixel}} \\
\mu_O^t &= \frac{\sum_O f(i,j)}{\text{number of object pixel}} T^{t+1} = \frac{\mu_B^t + \mu_O^t}{2}
\end{aligned}$$

If $T^{t+1} = T^t$, stop.

**Otsu Thresholding** This variant assumes a bimodal histogram and selects thresholds by histogram analysis. Thus, once the histogram with probability $P(i)$ in bin $i$ has been established, its fast. The basic idea is to find the threshold that minimizes the weighted intra-class variance:

$$\sigma^2_{intra}(t) = q_O(t)\sigma^2_O(t) + q_B(t)\sigma^2_B(t)$$

where $q_O(t) = \sum_{i=1}^{t} P(i)$ and $q_B(t) = \sum_{i=t+1}^{I} P(i)$ and the class means defined as $\mu_O(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_O(t)}$ and for background by analogy. Finally, the individual class variances are given as

$$\sigma^2_O(t) = \sum_{i=1}^{t} (i - \mu_O(t))^2 \frac{P(i)}{q_O(t)}$$

and the background variance by analogy. Now, we could actually stop here. All we need to do is just run through the full range of t values [1,256] and pick the value that minimizes $\sigma^2_{intra}$.

But the relationship between the intra-class and inter- class variances can be exploited to generate a recursion relation that permits a much faster calculation: For any given threshold, the total variance is the sum of the weighted intra-class variances and the inter-class cariance (which is the sum of weighted squared distances between the class means and the overall mean).

$$\sigma^2 = \sigma^2_{intra}(t) + \sigma^2_{inter}(t) = \sigma^2_{intra}(t) + q_O(t)(1 - q_O(t))(\mu_O(t) - \mu_B(t))2$$

Since the overall variance does not depend on a threshold, minimizing the intra-class variance is the same as maximizing inter-class variance. $\sigma^2_{inter}(t)$ can be computed more efficiently as compared to $\sigma^2_{intra}(t)$ by a recursion when running through the range of $t$ values:

- Initialisation: $q_O(1) = P(1)$, $\mu_O = 0$
- Recursion
  1. $q_O(t+1) = q_O(t) + P(t+1)$

2. $\mu_O(t+1) = \frac{q_O(t)\mu_O(t)+(t+1)P(t+1)}{q_O(t+1)}$
3. $\mu_B(t+1) = \frac{\mu - q_O(t+1)\mu_O(t+1)}{1-q_O(t+1)}$

**Thresholding in hierarchical data structures** uses a pyramidal data structure (see also section 2.6). Regions are identified in the low resolution image and are further refined in the higher resolutions. Two variants are possible:

1. Segmentation in the low-resolution image using thresholding; in the next higher resolution pixels close to the region border are re-assigned to object(s) or background, this is done successively until the highest resolution.
2. A *significant pixel detector* is applied in the lowest resolution image and determines pixels different to their neighbourhood. This is usually done with 3 x 3 masks which indicate a central pixel being different from their neighbourhood. Such pixels correspond to regions in full resolution. The corresponding image part in full resolution is thresholded using $T$ being between the gray-scale of the significant pixel and the average of the other 8-neighbours (this is done locally with different thresholds for different regions).

The advantage of hierarchical techniques is their higher robustness against noise, since the initial segmentations start with a significantly smoothed image.

## 6.4 Edge-based Techniques

Edge-based segmentation techniques are the oldest ones. But edge detection does not result in a segmented image. For this purpose, edge pixels need to be connected to *edge chains*, which correspond to region borders. The following three subsections are dedicated to this kind of segmentation.

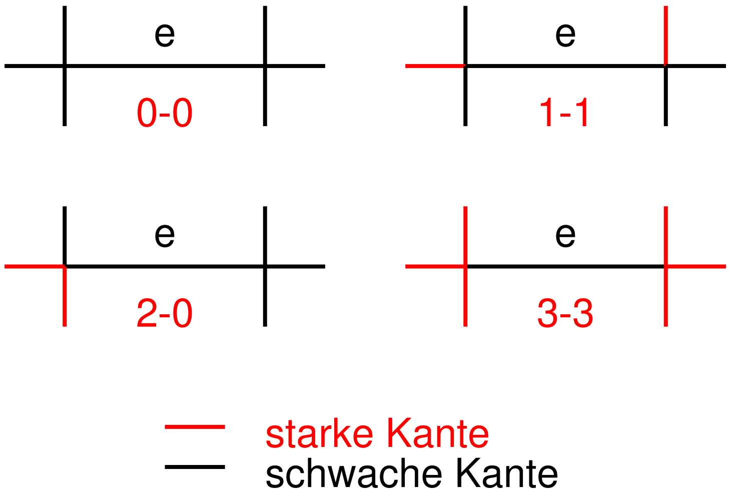
### 6.4.1 Thresholding of Edge Images

Small edge values correspond to non-significant gray-scale changes; these values can be excluded by a simple thresholding procedure. Further processing can be done by further excluding isolated edge pixels or edge-chains with a length below a certain length threshold.

### 6.4.2 Edge Relaxation

Edge thresholding is often impacted by noise which results in missing edge-chain parts to form complete region borders. Edge relaxation assesses edge property in the context of neighbouring pixels (compare thresholding with hysteresis). Under consideration of edge magnitude and edge continuation an iterative process increases or decreases the edge property. For this algorithm, edges are considered as *crack edges* (see section 2.3.3).

The edge property is investigated at both ends of an edge in all three possible edge continuation directions. Edge $e$ has a crossing at each side and there are three possible continuation direction (see figure 113). Each crossing is assessed according to its number of “leaving” edges and the form of the crossing. The initial edge property $c^1(e)$ is the normalised magnitude of the crack edge. Edge property converges from iteration to iteration to 0 or 1.



| | | | | | |
|---|---|---|---|---|---|
| 0-0 | isolated edge | - | 0-2 | dead end | - |
| 0-3 | dead end | - | | | |
| 0-1 | neutral | 0 | 2-2 | bridge | 0 |
| 2-3 | bridge | 0 | 3-3 | bridge | 0 |
| 1-2 | continuation | + | 1-3 | continuation | + |
| 1-1 | continuation | ++ | | | |

Figure 113: Edge properties

**Algorithm (2. and 3. are iterated)**

1. evaluate edge property $c^1(e)$ for all crack edges in the image
2. determine edge types in the neighbourhood and determine crossing types
3. update $c^{k+1}(e)$ for for each edge corresponding to its type and $c^k(e)$
4. stopping criterion (e.g. in case of convergence to 0 or 1)

**Assessment of crossing types**
Crossing is of type $i$, if $\text{type}(i) = \max_k((\text{type}(k)), k = 0, 1, 2, 3$

$$\text{type}(0) = (m-a)(m-b)(m-c) \qquad \text{type}(1) = a(m-b)(m-c)$$
$$\text{type}(2) = ab(m-c) \qquad \text{type}(3) = abc$$

| | |
|---|---|
| $a, b, c$ | … normalised values of neighbouring edges |
| $m$ | … $m = \max(a, b, c, q)$ |
| $q$ | … constant; $q \sim 0.1$ |

Example: $(a, b, c) = (0.5, 0.05, 0.05)$ is a type 1 crossing, $(0.3, 0.2, 0.2)$ is a type 3 one.

Similar results are obtained by counting the number of edges at a crossing above a threshold.

**Update Step**

**Increasing edge property** : $c^{k+1}(e) = min(1, c^k(e) + \delta)$

**Decreasing edge property** : $c^{k+1}(e) = max(0, c^k(e) - \delta)$

$\delta$ is typically selected from 0.1 - 0.3 (for strong and weak modification), simpler if only one value is used. In the version discussed so far, often only slow convergence is achieved.

**Improved Update Step**

$$c^{k+1}(e) = \begin{cases} 1 & c^{k+1}(e) > T_1 \\ 0 & c^{k+1}(e) \leq T_2 \end{cases}$$

### 6.4.3 Completing Edge Chains using Graph Search

| | |
|---|---|
| Initial situation: | Begin and end of an edge chain |
| Graph: | set of edges $x_i$ and paths $[x_i, x_j]$ connecting these edges |

We consider oriented and weighted paths and denote the weights as "costs".

Edge search is transformed into search for an optimal path in a weighted graph. We search for the best path connecting begin and end of an edge chain. We assume to have information about edge magnitude (gradient magnitude) $s(x)$ and edge orientation (gradient direction) $\phi(x)$. Each (edge-)pixel corresponds to a node in the graph, weigthed with $s(x)$. Two edges $x_i$ and $x_j$ can be connected by a path, if $\phi(x_i)$ and $\phi(x_j)$ fit together: $x_i$ has to be one of the three existing neighbours of $x_j$ in the direction $d \in [\phi(x_i) - \pi/4, \phi(x_j) + \pi/4]$ and $s(x_i)$ und $s(x_j) \geq T$.

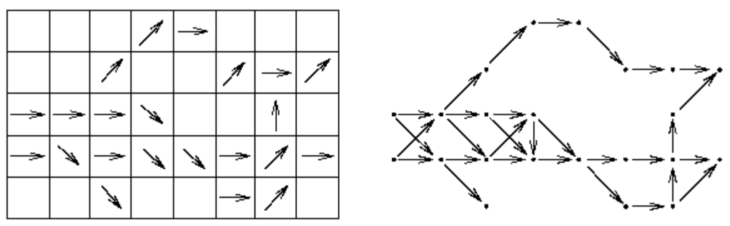


Figure 114: Graph representation of an edge image

Based on his set-up, classical graph search techniques can be applied. Let $x_A$ and $x_B$ be begin and end-point of the edge chain. We define an expansion technique plus a cost function $f(x_i)$, which allows to conduct an estimation of the costs of a path between $x_A$ and $x_B$ passing through $x_i$. The cost function has to be monotonic with respect to path length and it has to be possible to partition the computation of the function among several path parts.

| | |
|---|---|
| $g(x_i)$ | costs from $x_A$ to $x_i$ |
| | sum of the costs of the nodes in the path connecting $x_A$ and $x_i$ |
| $h(x_i)$ | costs from $x_i$ to $x_B$ (estimation) |

**Nielson's A-Algorithm** (heuristic graph search)

1. expand $x_A$ and put all successors into an OPEN-List with pointers back to $x_A$; compute costs for all nodes.
2. if OPEN-List is empty, the algorithm failed.
   otherwise: determine $x_i$ in the list with the lowest costs $f(x_i)$ and remove this node. If $x_i = x_B$ follow the pointers to identify the best path and stop.
3. no "stop" occured in 2. Expand $x_i$ and put the successors into the OPEN-List with pointers back to $x_i$; compute their costs; go to 2.

It is important to include a strategy against the occurence of loops in the algorithm. The estimation $\hat{h}(x_i)$ of $h(x_i)$ has significant influence to the behaviour of the search process (search can be accelerated - less precise - or search can degenerate to full search).

**Variants**

$\hat{h}(x_i) = 0$ : No heuristics is included and the search degenerates into a breadth-first search. Heuristic methods do not guarantee to find the optimal result but they are faster.

$\hat{h}(x_i) > h(x_i)$ : The algorithm is fast, but the minimal cost result cannot be guaranteed.

$\hat{h}(x_i) = h(x_i)$ : The search is able to identify the path with lowest costs while using a minimal number of expanded nodes. In general, we find that the number of expandend nodes is the smaller, the closer $\hat{h}(x_i)$ is to $h(x_i)$.

$\hat{h}(x_i) \leq h(x_i)$ : The search identifies the path with lowest costs, however, for each part of the path we find that actual costs are larger than estimated costs.

**Branch and Bound algorithms** maximal admissible costs are defined and more expensive paths are no longer considered in the search.

**Multiresolution Processing** Ideas described so far are applied to low-resolution-image data first, and transferred to higher resolution subsequently (usually in a prediction - correction framework).

Applicable cost functions:

- Edge magnitude: high magnitude $\rightarrow$ reliable edge $\rightarrow$ small costs (e.g. direct costs: difference to largest edge magnitude in the image).
- Curvature: Difference of edge orientations
- Distance to end-point of edge chain

- Distance to known or estimated or conjectured position of the edge chain

**Dynamic Programming**

Background: Bellmann's principle of optimimality – independent of the path leading to node E, there is an optimal path connecting E and the end point. In other words: If the optimal path connecting begin and end point point passes through E, also the partial paths begin point → E and E → end point have to be optimal.

1. Construct the graph and the rating (based on weights) of all partial paths between two layers.
2. In each layer, determine for each node E the lowest-cost partial path connecting the preceeding layer to E.
3. Determine the lowest-cost node in the final layer.
4. Backtracking along the identified optimal partial paths identifies the overall optimal path.

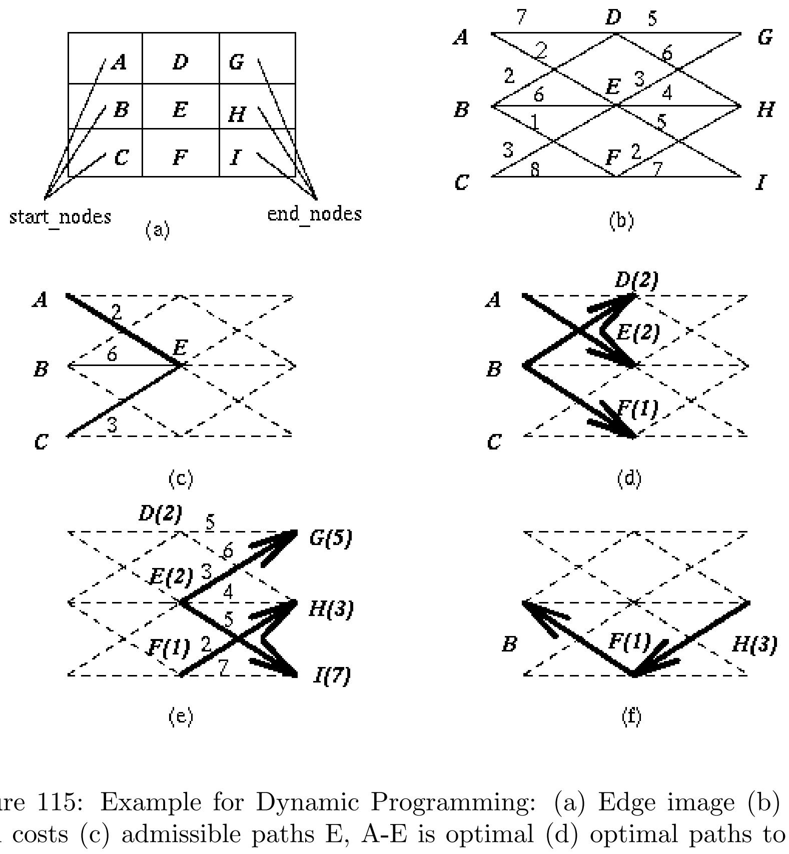


Figure 115: Example for Dynamic Programming: (a) Edge image (b) Graph with costs (c) admissible paths E, A-E is optimal (d) optimal paths to D,E,F (e) optimal paths to G,H,I (f) Backtracking from H determines lowest-cost path

### 6.4.4 Active Contours - Snakes / Level Set Segmentation

These techniques iteratively adapt a closed curve to image content by optimising energy functionals: External term (attracts the contour toward the closest image edge, obviously gradient information is used) and internal terms (force the contour to be continuous and smooth). Important questions are how to select the initial curve and how to weight the different terms in the functional. [For AISP, more details in the "Medical Imaging" lecture].

## 6.5 Region-based Techniques

Such techniques are primary applied to noisy images since under such conditions edge detection is difficult and results are not very reliable. Also, in circumstances where the major difference among pixels of different objects is not their differing luminance (e.g. different structure - texture), edge-based techniques may fail due to lacking contrast. The strategy is to partition an image into regions of maximal homogeneity.

**Homogeneity**
Homogeneity is often defined based on gray-values (e.g. average gray-value, shape of a local histogram) or uses textur properties etc.

Target regions exhibit the follwoing property (apart from their zero intersection):

$$\begin{aligned} H(R_i) &= \text{True} \qquad & i &= 1, \ldots, S \\ H(R_i \cup R_j) &= \text{False} \qquad & i &\neq j \text{and} R_i \text{ adjacent to } R_j \end{aligned}$$

$S \ldots$ number of regions, $H(R_i)$ is a binary evaluation of homogeneity of $R_i$; this means that regions are homogeneous and maximal (i.e. if they would be larger, homogeneity is lost).

In the following sections we discuss region-based techniques.

### 6.5.1 Region Growing

Starting from seed pixels, new (adjacent) candidates are investigated and added to the region, in case homogeneity is maintained.

1. Choose seed pixel(s).
2. Check neighbouring pixels following a chosen strategy and add them to the region if the are "similar" to the seed.
3. Repeat step 2) for the newly added pixels; stop if no more epixels can be added.

Criteria how to define similarity (or homogeneity of the region) include average intensity, variance, colour (histograms), texture, motion, shape, etc. The selection of seed points is highly application dependent and can be done similar to marker selection in watershed segmentation (see next section), the simplest choice is to select intensity / colour values corresponding to the highest peaks in the histograms.

### 6.5.2 Region Merging

1. segment image into small regions satisfying the homogeneity requirement
2. define criteria to merge those regionens (*merging*)
3. apply merging until it is no longer possible

Different technique differ in terms of their initial segmentations and the different criteria for homogeneity. An improvement is the additional employment of edge inforation: neighbouring regions are merged if a significant share of their common border consists of weak edges. The result of edge relaxation can be used to determine if an edge is weak or not.

### 6.5.3 Region Splitting

Contrasting to *Region Merging* we start with the entire image which usually does not satisfy the homogeneity criterum. The image is split into regions satisfying this constraint.

Remark: Merging is not dual to splitting even if the same hoogeneity criterium is used ! See splitting and merging in figure 116 applied to a chess board, where identical average gray-value is used as criterium to rate homogeneity. In case of splitting the criterium is a medium gray for all for image quadrants (so the criterium is fulfilled and no splitting is applied), in case of merging it is black or white until the size of the chess-board pattern is reached (i.e. we result in black and white regions corresponding to the fileds of the board).

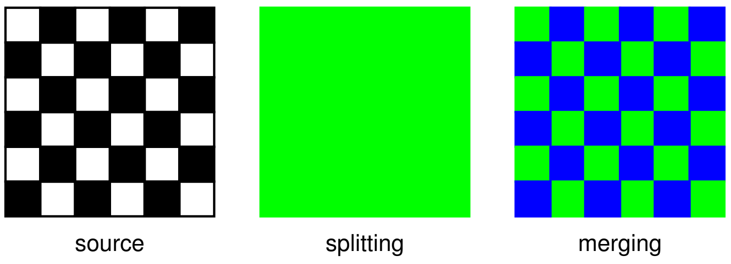


Figure 116: splitting/merging

**Application**
In many cases a combination of splitting and subsequent merging is applied – the datastructure employed is often a quadtree (*split and merge*):

1. Define an initial segmentation into regions, a criterium for homogeneity and a pyramidal data structure.
2. In case a region in the data structure is not homogeneous, it is split into its four children regions; In case four regions corresponding to the same parent can be merged, they are merged. If no further region can be processed, GOTO 3)
3. In case two neighbouring regions (either in different levels of the pyramid or with different parent nodes) can be merged according the critierium for homogeneity the are merged.
4. Regions too small are merged with the most similar neighbouring region.

### 6.5.4 Template Matching

Known objects are identified by computing the difference to given *templates* – thus, a template needs to be generated representing the object of interest as well and as general as possible.

### 6.5.5 Watershed Segmentation

Watershed segementation uses techniques from morphological image processing and is therefore discussed in the follwoing section as final application of morphological operations.

### 6.5.6 Mean Shift Segmentation

Image segmentation can be viewed as a clustering task, i.e. "similar" pixels are clustered into one region. This leads to the following procedure:

1. Represent each pixel in the image with a vector (e.g. intensity, colors, colour and location, texture descriptors, etc.)
2. Choose distance weights (which vector component is more important than others, e.g. color vs. location)
3. Apply k-means clustering
4. Pixels belong to the segment corresponding to cluster centers

Different representations of pixels lead to different segmentation results (e.g. if location is not at all employed, clusters are not spatially coherent).

**K-means Clustering**: Given a set of pixels $(x_1, \ldots, x_n)$ and represent each pixel by its associated vector. K-means clustering aims to partition the $n$ pixels into $K$ sets ($K \leq n$) $S = \{S_1, S_2, \ldots, S_K\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_S \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

with $\mu_i$ the mean or centroid of pixels in cluster $S_i$. This is usually done in two steps. In the **assignment step**, each pixel is assigned to the cluster whose mean yields the least WCSS (geometrically, this corresponds to a partitioning of the pixels according to the *Voronoi diagram* generated by the means). In the **update step**, the new means / centroids in the new clusters are computed.

This approach has some severe drawbacks:

- Sensitive to initialisation (how to choose initial $\mu_i$ – e.g. random, uniform partitioning – similar to the question how to select seeds in region growing).
- $K$ - the number of image segments has to be chosen a priori.
- Sensitive to outliers.
- A key limitation of K-means is its cluster model. The concept is based on spherical clusters that are separable in a way so that the mean value converges towards the cluster center.

The **mean shift** algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. The idea is that clusters are places where data points (i.e. pixels) tend to be close together in feature space. Thus, instead of initialising cluster centers and selecting the number of clusters, the mean shift algorithm seeks *modes* or local maxima of density in the feature space.

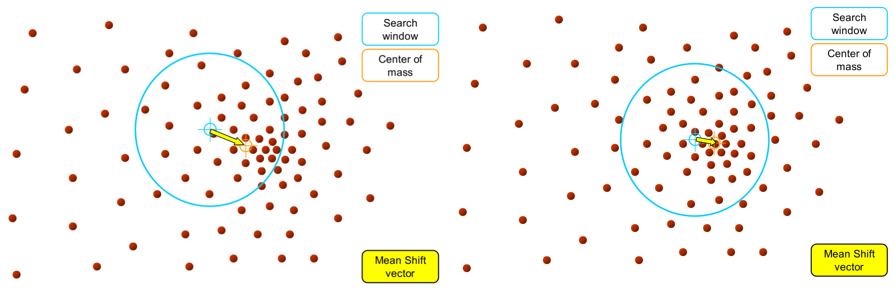


Figure 117: Mean-shift procedure in feature space

The actual mean shift procedure selects a pixel, applies a window in feature space around the pixels' feature and subsequently, the mean of the pixels' neighbouring feature vectors is computed. The center of the window is shifted to the position of the mean (centroid) and the procedure is iterated until the mean does not change its position. So a local maximum – a mode – is found (see Fig. 117).

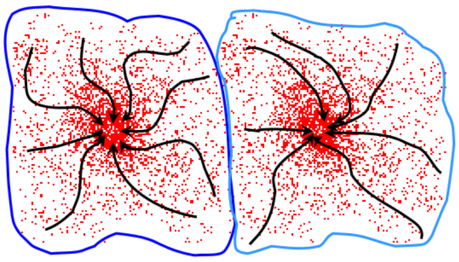


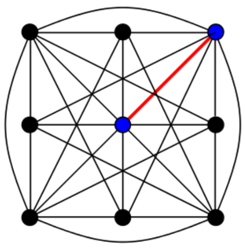
Figure 118: Constructing basins of attraction

The set of all feature vectors that converge to the same mode defines the basin of attraction of that mode. The points – pixels – which are in the same basin of attraction are associated with the same cluster and form image regions (Fig. 118).

### 6.5.7 Graph Cut Segmentation

We represent images as graphs by (see Fig. 119)

- assigning a vertex to each pixel,
- defining edges between neighbouring pixels,
- weighting edges according to the similarity of pixels (*affinity* of vertices), i.e. distance between representation vectors.



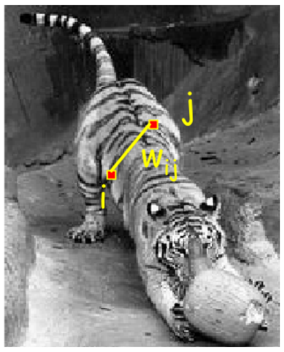


Figure 119: Representing images as graphs.

Segmenting an image thus corresponds to cutting this graph into segments, i.e. removing edges that cross between graph parts corresponding to similar regions (see Fig. 120). Obviously, it is easiest to break links that have low affinity (since similar pixels should be in the same segments and dissimilar pixels should be in different segments).

**Graph Cut** is the set of edges the removal of which makes a graph disconnected and of course, a graph cut provides a segmentation. The cost of a cut

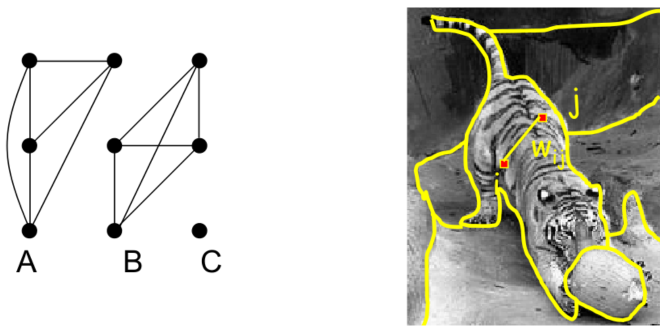


Figure 120: Segmenting images as graph partitioning.

is determined by the sum of weights of the cut edges. The minumum cut (with lowest costs) tends to cut off very small, isolated components (see Fig. 121), a problem which is resolved by the normalised cut: Here the cost of an edge connecting A and B is normalised by the costs of all edges involving A and by the costs of all edges involving B. Minimizing these costs leads to more robust graph cuts.

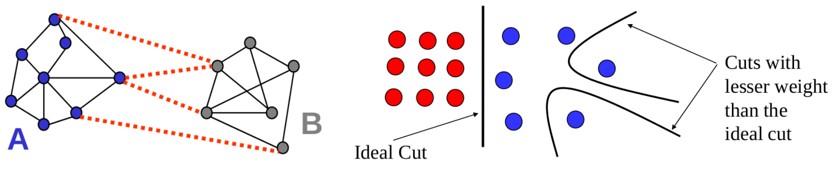


Figure 121: Problems with minimum cut.

## 7 Morphological Image Processing

*Disclaimer: This section is a shortened and edited version of the section 18.7 Binary Image Processing from the book* ***Fundamentals of Digital Image Processing*** *pages 470 to 475.*

Binary images—those having only two gray levels—constitute an important subset of digital images. A binary image (e.g., a silhouette or an outline) normally results from an image segmentation operation. If the initial segmentation is not completely satisfactory, some form of processing done on the binary image can often improve the situation.

Many of the processes discussed in this section can be implemented as $3 \times 3$ neighborhood operations. In a binary image, any pixel, together with its neighbors, represents nine bit of information. Thus, there are only $2^9 = 512$ possible configurations for a $3 \times 3$ neighborhood in a binary image.

Convolution of a binary image with a $3 \times 3$ kernel (see figure 122) generates a nine-bit (512-gray-level) image in which the gray level of each pixel specifies the configuration of the $3 \times 3$ binary neighborhood centered on that point.

Neighborhood operations thus can be implemented with a 512-entry look-up table with one-bit output.

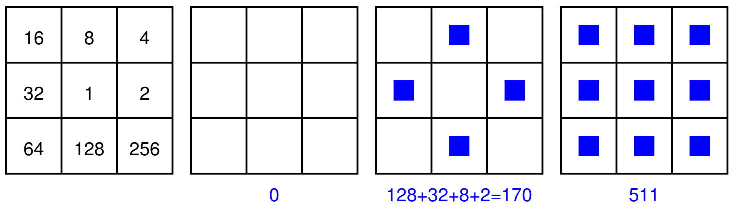


Figure 122: Binary neighborhood encoding

This approach can be used to implement a logical operation called a *hit-or-miss transformation.* The look-up table is loaded to search for a particular pattern—for example, all nine pixels being black. The output is one or zero, depending on whether the neighborhood matches the mask. If, whenever the pattern is matched (a hit), the central pixel is set to white and the central pixel of all other configurations is left unchanged (a miss), the operation would reduce solid objects to their outlines by eliminating interior points.

## 7.1 Morphological Image Processing

A powerful set of binary image processing operations developed from a set-theoretical approach comes under the heading of *mathematical morphology.* Although the basic operations are simple, they and their variants can be concatenated to produce much more complex effects. Furthermore, they are amenable to a look-up table implementation in relatively simple hardware for fast *pipeline processing.* While commonly used on binary images, this approach can be extended to gray-scale images as well.

In general case, morphological image processing operates by passing a *structuring element* over the image in an activity similar to convolution (see figure 123). Like the convolution kernel, the structuring element can be of any size, and it can contain any complement of 1's and 0's. At each pixel position, a specified logical operation is performed between the structuring element and the underlying binary image. The binary result of that logical operation is stored in the output image at that pixel position. The effect created depends upon the size and content of the structuring element and upon the nature of the logical operation.

For this introduction to the subject, we concentrate on the simplest case, namely, the use of a basic $3 \times 3$ structuring element containing all 1's. With this restriction, it is the logical operation that determines the outcome.

### 7.1.1 Set Theory Nomenclature

In the language of morphological processing, both the binary image, **B**, and the structuring element, **S**, are sets defined on a two-dimensional Cartesian grid, where the 1's are the elements of those sets.

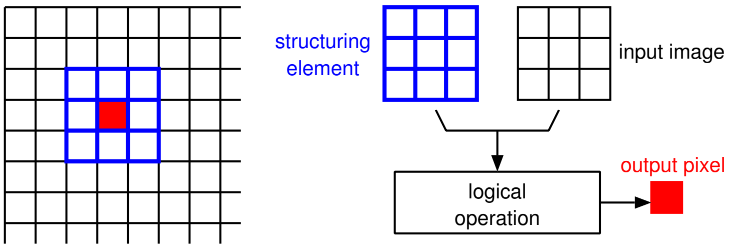


Figure 123: Morphological image processing

We denote by $\mathbf{S}_{xy}$ the structuring element after it has been translated so that its origin is located at the point $(x, y)$. The output of a morphological operation is another set, and the operation can be specified by a set-theoretical equation.

### 7.1.2 Erosion and Dilation

The basic morphological operations are erosion and dilation (see figure 124). By definition, a boundary point is a pixel that is located inside an object, but that has at least one neighbor outside the object.

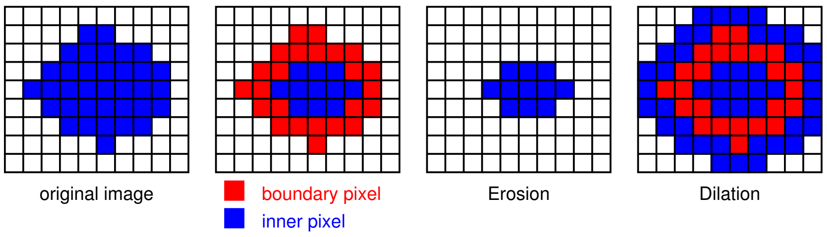


Figure 124: Erosion and dilation

Simple **erosion** is the process of eliminating all the boundary points from an object, leaving the object smaller in area by one pixel all around its perimeter. If the object is circular, its diameter decreases by two pixels with each erosion. If it narrows to less than three pixels thick at any point, it will become disconnected (into two objects) at that point. Objects no more than two pixels thick in any direction are eliminated. Erosion is useful for removing from a segmented image objects that are too small to be of interest.

General erosion is defined by

$$\mathbf{E} = \mathbf{B} \otimes \mathbf{S} = \{x, y | \mathbf{S}_{xy} \subseteq \mathbf{B}\} \tag{28}$$

The binary image $\mathbf{E}$ that results from eroding $\mathbf{B}$ by $\mathbf{S}$ is the set of points $(x, y)$ such that if $\mathbf{S}$ is translated so that its origin is located at $(x, y)$, then it is completely contained within $\mathbf{B}$. With the basic $3 \times 3$ structuring element, general erosion reduces to simple erosion.

Simple **dilation** is the process of incorporating into the object all the background points that touch it, leaving it larger in area by that amount. If the object is circular, its diameter increases by two pixels with each dilation. If the two objects are seperated by less than three pixels at any point, they will become connected (merged into one object) at that point. Dilation is useful for filling holes in segmented objects.

General dilation is defined by

$$\mathbf{D} = \mathbf{B} \oplus \mathbf{S} = \{x, y | \mathbf{S}_{xy} \cap \mathbf{B} \neq \emptyset\} \tag{29}$$

The binary image **B** that results from dilating **B** by **S** is the set of points $(x, y)$ such that if **S** is translated so that its origin is located at $(x, y)$, then its intersection with **B** is not empty. With the basic $3 \times 3$ structuring element, this reduces to simple dilation.

### 7.1.3 Opening and Closing

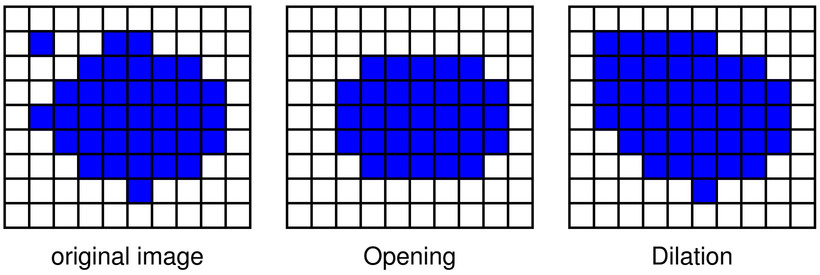


Figure 125: Opening and closing (auch in der Figure !)

The process of erosion followed by dilation is called **opening**. It has the effect of eliminating small thin objects, breaking objects at thin points, and generally smoothing the boundaries of larger objects without significantly changing their area. Opening is defined by

$$\mathbf{B} \circ \mathbf{S} = (\mathbf{B} \otimes \mathbf{S}) \oplus \mathbf{S} \tag{30}$$

The process of dilation followed by erosion is called **closing**. It has the effect of filling small and thin holes in objects, connecting nearby objects, and generally smoothing the boundaries of objects without significantly changing their area. Closing is defined by

$$\mathbf{B} \bullet \mathbf{S} = (\mathbf{B} \oplus \mathbf{S}) \otimes \mathbf{S} \tag{31}$$

Often, when noisy images are segmented by thresholding, the resulting boundaries are quite ragged, the objects have false holes, and the background is peppered with small noise objects. Successive openings or closings can improve the situation markedly. Sometimes several iterations of erosion, followed by the same number of dilations, produces the desired effect.

## 7.2 Shrinking

When erosion is implemented in such a way that single-pixel objects are left intact, the process is called shrinking. This is useful when the total object count must be preserved.

Shrinking can be used iteratively to develop a size distribution for a binary image containing approximately circular objects. It is run alternately with a $3 \times 3$ operator that counts the number of single-pixel objects in the image. With each pass, the radius is reduced by one pixel, and more of the objects shrink to single-pixel size. Recording the count at each iteration gives the cumulative distribution of object size. Highly noncircular objects (e.g., dumbbell-shaped objects) may break up while shrinking, so this technique has its restrictions.

## 7.3 Thinning

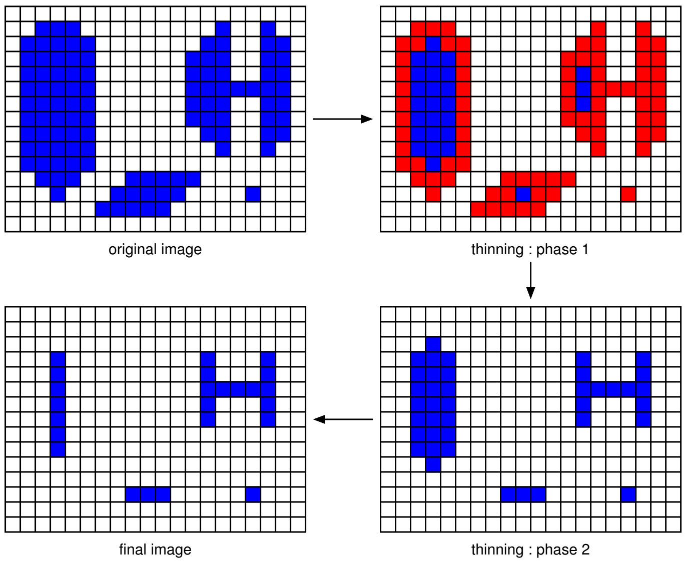


Figure 126: Thinning

Erosion can be programmed as a two-step process that will not break objects. The first step is a normal erosion, but it is conditional; that is, pixels are marked as candidates for removal, but are not actually eliminated. In the second pass, those candidates that can be removed without destroying connectivity are eliminated, while those that cannot are retained. Each pass is a $3 \times 3$ neighborhood operation that can be implemented as a table-lookup operation.

Thinning reduces a curvilinear object to a single-pixel-wide line. showing its topology graphically (see figure 126).

## 7.4 Skeletonization

An operation realted to thinning is skeletonization, also known as *medial axis transform* or the *grass-fire technique*. The medial axis is the locus of the centers of all the circles that are tangent to the boundardy of the object at two or more disjoint points. Skeletonization is seldom implemented, however, by actually fitting circles inside the object.

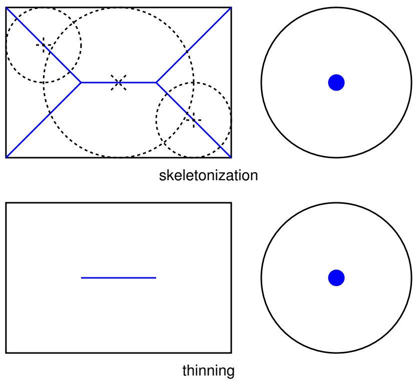


Figure 127: Skeletonization

Conceptually, the medial axis can be thought of as being formed in the following way. Imagine that a patch of grass, in shape of the object, is set on fire all around the periphery at once. As the fire progresses inward, the locus of points where advancing fire lines meet is the medial axis.

Skeletonization can be implemented with a two-pass conditional erosion, as with thinning. The rule for deleting pixels, however, is slightly different (see figure 127).

## 7.5 Pruning

Often, the thinning or skeletonization process will leave spurs on the resulting figure. These are short branches having an endpoint located within three or so pixels of an intersection.

Spurs result from single-pixel-sized undulations in the boundary that give rise to a short branch. They can be removed by a series of $3 \times 3$ operations that remove endpoints (thereby shortening all the branches), followed by reconstruction of the branches that still exist. A three-pixel spur, for example, disappears after three iterations of removing endpoints. Not having an endpoint to grow back from, the spur is not reconstructed.

## 7.6 Thickening

Dilation can be implemented so as not to merge nearby objects. This can be done in two passes, similarly to thinning. An alternative is to complement the image and use the thinning operation on the background. In fact, each of the variants of erosion has a companion dilation-type operation obtained when it is run on a complemented image.

Some segmentation techniques tend to fit rather tight boundaries to objects so as to avoid erroneously merging them. Often, the best boundary for isolating objects is too tight for subsequent measurement. Thickening can correct this by enlarging the boundaries without merging separate objects.

## 7.7 Application: Watershed Segmentation

A good (animated) visualisation and several examples may be found at:
`http://cmm.ensmp.fr/~beucher/wtshed.html`

Watersheds separate different (water)basins – in order to be able to transfer this notion into image processing context, the image is interpreted as three dimensional structure: Luminance values are interpreted as height measure (elevation at the corresponding image coordinates). In many cases a gradient image is used for further processing. Region borders (i.e. edge chains) correspond to "high valued" edge chains and inner areas with low gradient correspond to basins (see Fig. 128).

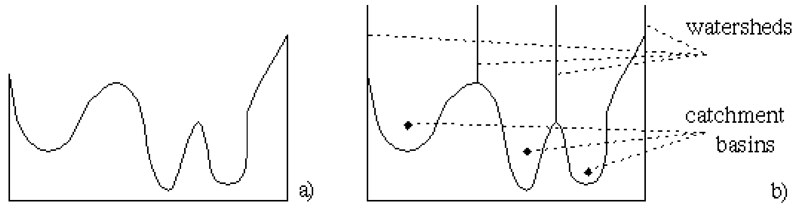


Figure 128: Principles: Watersheds and basins

Basins are homogeneous in the sense that all pixels belonging to a catchment basin are connected to the minimum value of the basin by a path, the pixel of which are monotonically decreasing in direction to the minumum. Basins represent the regions of the segemented image, the watersheds represent region borders.

There are two differnt approaches to watershed transform:

1. The first approach determines a "downstream" path to a minimum for each pixel. A catchment basin is defined as the set of pixels the path of which leads to the same minimum. A problem of this approach is the unique determination of the path (which can be facilitated by local gradients in the contineous case).

2. The second appraoch is dual to the first one and uses "flooding": catchment basins are flooded from below (assume that holes are at the location of minima in the 3D surface , when submerging the surface, water enters through the holes). As soon as two catchment basins would be fused due to rising water, a dam is construted to prevent this. The value of the dam pixels is set to the maximum value of the image.

Subsequently, we focus onto the second technique. For preprocessing, pixels are sorted according to their gray-scale, the gray-scale histogram is computed, and a list of pointers is built pointing at pixels with gray-scale $h$. In this manner, all pixels with a specific gray-scale can be addressed efficiently. Assume that flooding has been propagated until gray-scale $k$. Each pixel with gray-scale $\leq k$ has been uniquely assigned to a basin and carries its label. In the next step all pixels with gray-scale $k + 1$ are processed. A pixel with this gray scale can be assigned to basin $l$ in case at least one direct neighbour carries label $l$. In order to determine membership to a basin, zones of influence are defined: The zone of influence of a basin $l$ are the positions of the not-assigned but connected pixels with gray-scale $k+1$, the distance to $l$ of which is smaller than to any other basin.See Fig. 129 for a visualisation.

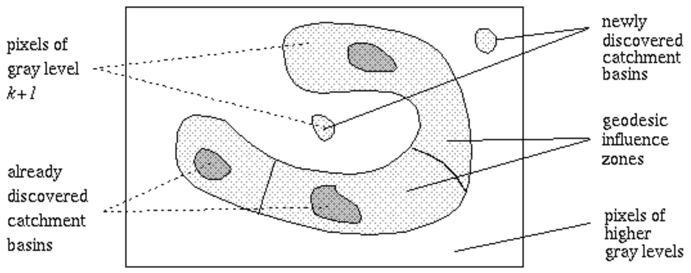


Figure 129: Zones of influence of basins

All pixels with gray-scale $k+1$ belonging to the influence zone of the basin $l$ are assigned the label $l$, which means that catchment basins grow. Not-yet assigned pixels are processed successively, pixels without label correspond to new basins and get a new label. The border separating the catchment basins ae the watersheds. Fig. 130 illustrates the entire procedure.

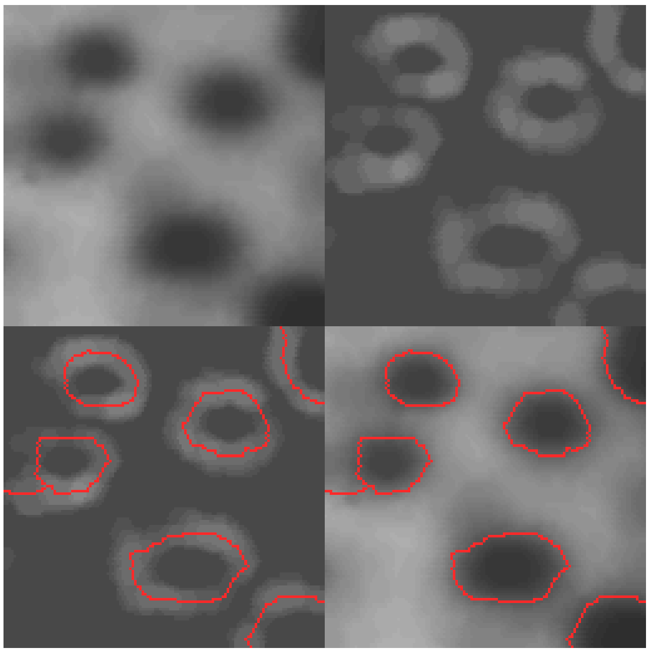


Figure 130: Original, gradient image, watersheds, original with watersheds

Note that so far, we have not described how to explicitly construct dams – this will be explained subsequently using morphological operators. When applying watershed segmentation as described so far, we result in significant oversegmentation quite often (i.e. too may regions are formed, see Figs. 131 and 133.c), Since the number of minima is simply too high (e.g. caused by noise).

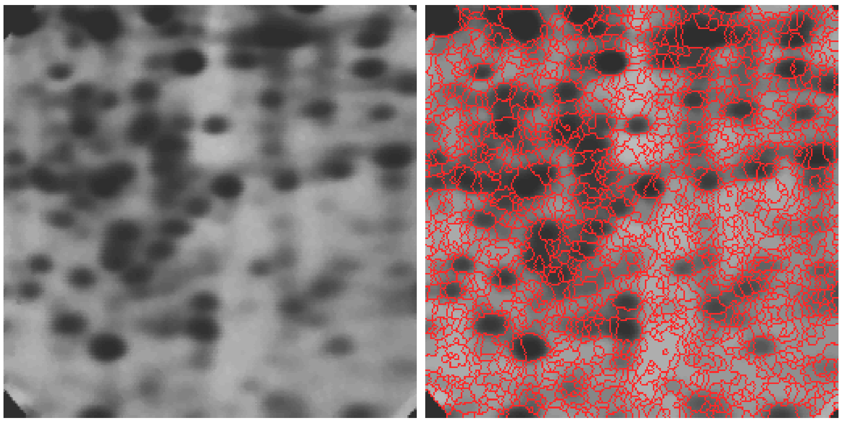


Figure 131: Oversegmentation

In order to limit this effect, the follwoing strategy can be applied:

- Image smoothing (e.g. Gauss filtering with large $\sigma$)
- Marker: Only "internal markers" are accepted as local minima, i.e. contigous regions with identical gray-scale surrounded by pixels with larger value.

Fig. 132 left shows internal markers. Subsequently the watershed algorithm is applied to the image. The resulting watershed lines are denoted as "external markers", which partition the image into regions, where each region contains a single object with its associated background. Now we can apply a watershed procedure or thresholding to each region to arrive at the desired segmentation. Fig. 132 right and 133.d show corresponding results. As an additional measure, the number of internal markers can be limited or a minimum size can be required.

**Dam construction**

In case of the requirement of an explicit dam construction process during flooding, the flooding step $n-1$ right before a fusion of two catchment basins is taken as the initial stage for dam construction (see the two black regions in Fig. 134). The fused region after step $n$ is denoted as $q$ (depicted in white). Subsequently, dilation is applied to the two black regions using a constant 1 3 x 3 structuring element, satisfying two conditions:

1. The center of the structuring element resides in $q$.
2. Dilation is only applied without a fusion of the two regions.

In the example, the first dilation step can be conducted without any problems, the two black regions are enlarged consistently. In the second dilation run

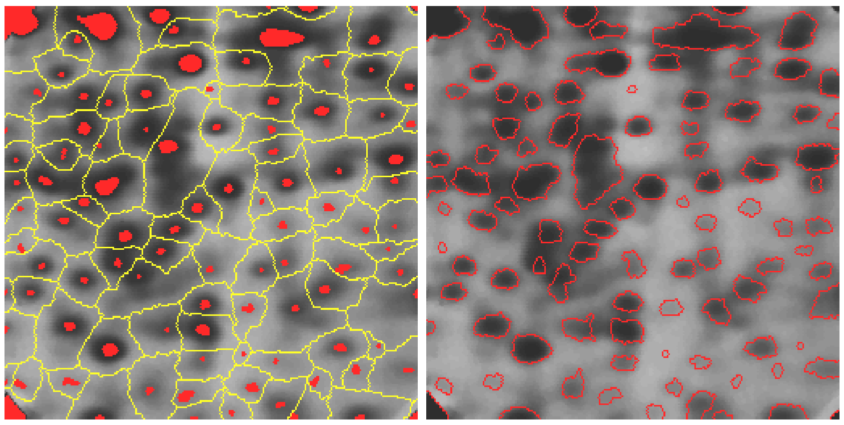


Figure 132: Watersheds with internal and external markers

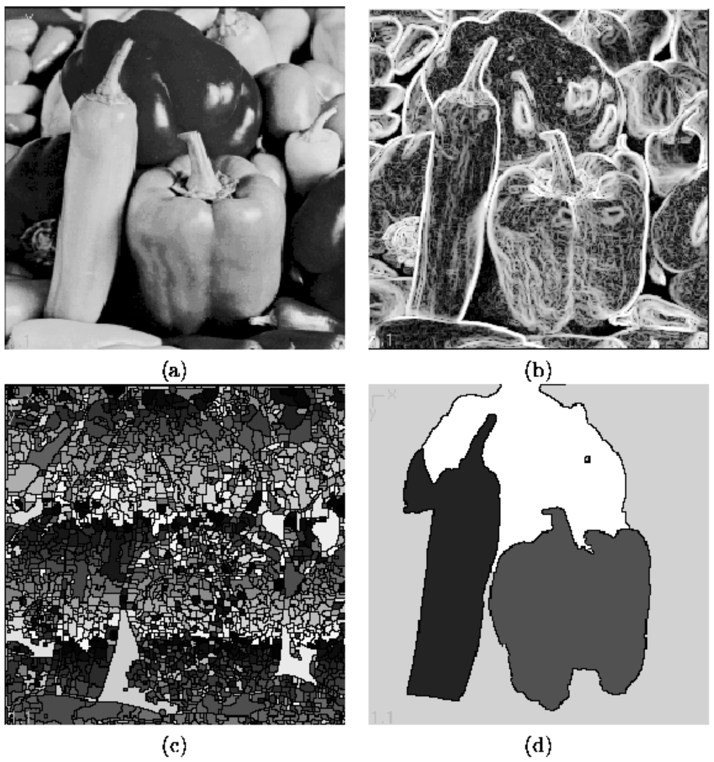


Figure 133: Example: All variants

several points do not satisfy the first condition, that is why the perimeter curve is disconnected. Points satisfying the first condition but not the second one are defined to be dam points. These are set to the maximal luminance value in order not to get over-flooded again. Then, flodding is continued.

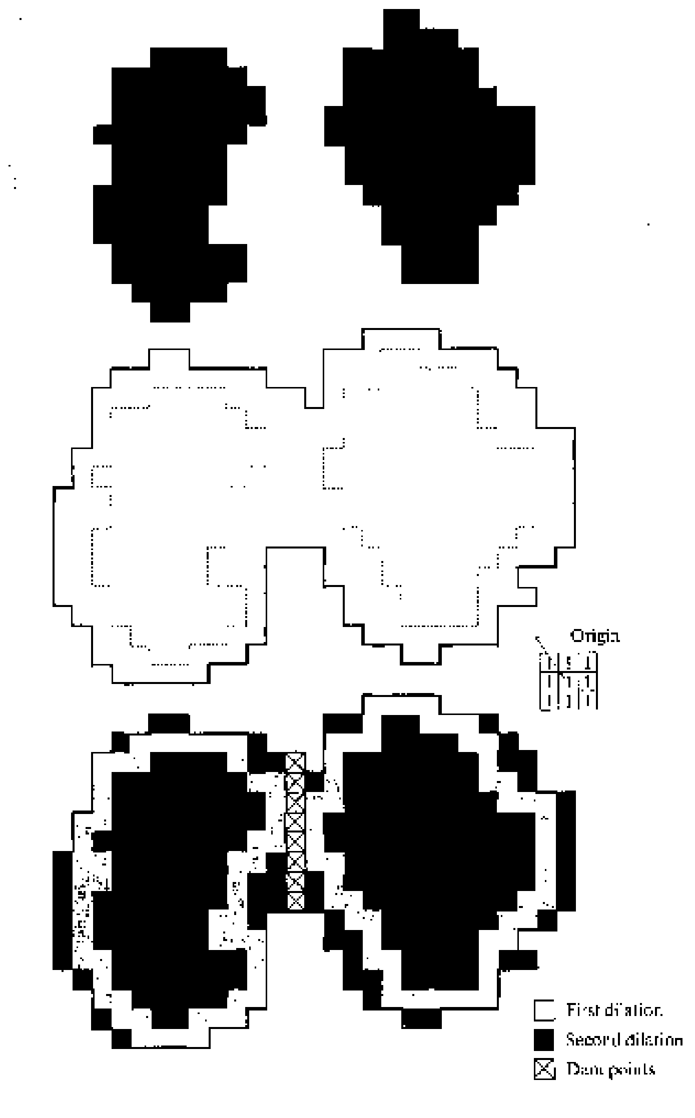


Figure 134: Dam construction with dilation

## 8 Image Formation

The image formation process can be structured into exposure and autofocus control which physically (i.e. optically, electronically, and mechanically) influence the captured visual information coming out from the sensor and the subsequent color image processing pipeline, which applies image processing operations onto the captured data.

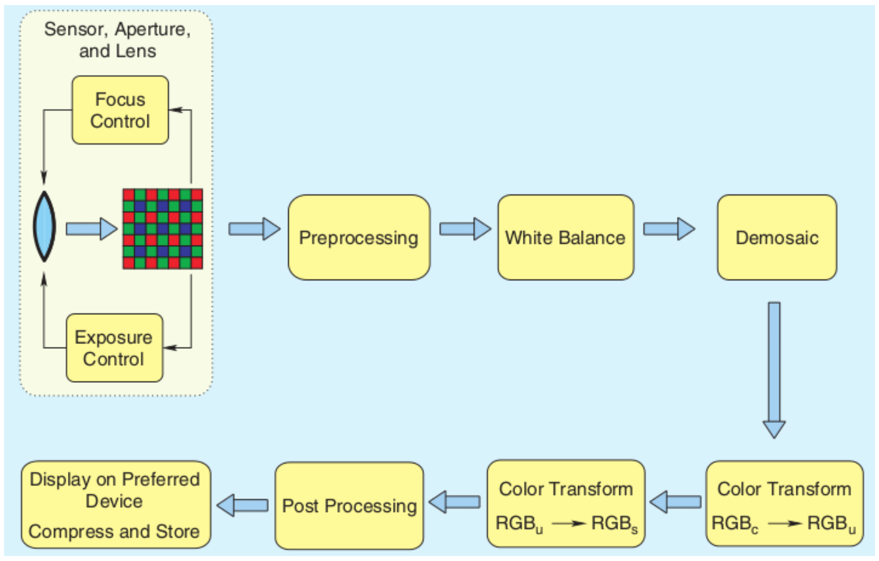

Sensor, Aperture, and Lens
Focus Control
Exposure Control
Preprocessing
White Balance
Demosaic
Color Transform $RGB_c \rightarrow RGB_u$
Color Transform $RGB_u \rightarrow RGB_s$
Post Processing
Display on Preferred Device
Compress and Store

Figure 135: Color Imaging Pipeline: Coarse View.

## 8.1 Exposure & Autofocus

Central aspects of image quality are contrast and sharpness. While both aspects can be improved by image enhancement operations, primarily their properties should be optimised in the image acquisition process. This is done by exposure and focus control mechanisms in the camera.

### 8.1.1 Exposure

Exposure is controled by the "exposure triangle" where each item controls exposure differently:

- shutter speed (controls the duration of the exposure),
- aperture (controls the area over which light can enter the camera), and
- ISO speed (controls the sensitivity of the camera's sensor),

while scene luminance defines the required exposure. One can therefore use many combinations of the above three settings to achieve the same exposure. The key, however, is knowing which trade-offs to make, since each setting also influences other image properties. For example, aperture affects depth of field, shutter speed affects motion blur and ISO speed affects image noise. Shutter speed can be controled with a mechanical shutter or electronically, aperture is controled by the camera's iris, and ISO speed is adjusted by either varying the amplification applied to the sensors analog output signal before analog-to-digital (A/D) conversion or by remapping e.g. 12 bits worth of sensor CCD output onto 8 bits of digital output in the camera's A/D converter. In any case, noise is being amplified and even added by the first strategy.

Exposure control usually requires characterization of the brightness (or intensity) of the image: an over- or underexposed image will greatly affect output colors. Depending on the measured energy in the sensor, the exposure control system changes the settings in the exposure triangle. Both the exposure and focus controls may be based on either the actual luminance component derived from the complete RGB image or simply the green channel data, which is a good estimate of the luminance signal.

For determining the brightness of the image, it is usually divided into blocks and the average luminance signal is measured in each one of these blocks. The most common method is a centre-weighted average metering where luminance is averaged over all blocks while assigning more weight to the central 60 – 80 % of the image. Other metering approaches include spot metering (only central image parts are being used) and matrix metering (using a honeycomb configuration to identify objects and their luminance). Also more recently, face detection is employed to identify area of specific interest (i.e. faces) to evaluate luminance and optimise exposure for such areas. The exposure control then tries to change the exposure so that metring results fit a middle grey tone; a so called "18% grey".

An alternative to fitting the metering results to a specific average luminance value is to explicitly focus onto the luminance value distribution (in image regions or the entire image) by considering the image histogram. The histogram represents the dynamic range of the sensor and can be partitioned into e.g. 5 equally sized bins. An underexposed image will be leaning to the left (provided that left histogram regions represent dark colours), while an overexposed image will be leaning to the right in the histogram. Image details disappear in over- and underexposed images, hence, we want as much as possible of the image to appear in the middle region of the histogram. This can be quantified by computing the mean sample value (MSV), which determines the balance of the tonal distribution in the image:

$$MSV = \frac{\sum_{i=0}^{4}(i+1)x_i}{\sum_{i=0}^{4} x_i}$$

where $x_i$ is the number of pixels in histogram region $i$. Thus, the image is correctly exposed when $MSV \approx 2.5$.

The distribution of luminance values as determined in the metering process can also be combined to form a measure of exposure based on the type of scene being imaged: backlit or frontlit scene or a nature shot. In a typical image (nature shot), average luminance values are uniformly or randomly distributed across the scene. Backlit or frontlit scenes may be distinguished by measuring the difference between the average luminance signal in the central area A and background area B. If the image is excessively frontlit, the average luminance in region A will be much higher than that in region B, and vice versa in the case of a backlit scene. The exposure is controlled subsequently so as to maintain the difference between the average signals in these two areas, an estimate of the object-background contrast.

All metering techniques measure the light reflected from the scene and assume all tones within the scene that they are metering to average out to a mid-grey tone (which might not be true). If the scene has a lot of light tones, the camera will underexpose the image. The camera's meter gives an exposure reading that renders the light tones as grey, and this results in underexposure (to correct this, there are exposure correction settings for taking images e.g. in the snow).

Outdoor images (and many indoor ones as well) taken with typical cameras suffer from the problem of limited dynamic range in the case of an excessively backlit or frontlit scene. Dynamic range refers to the contrast ratio between the brightest pixel and the darkest pixel in an image. The human visual system (HVS) can adapt to about four orders of magnitude in contrast ratio, while the sRGB system and typical computer monitors and television sets have a dynamic range of about two orders of magnitude. This leads to spatial detail in darker areas becoming indistinguishable from black and spatial detail in bright areas become indistinguishable from white.

High dynamic range (HDR) solves this problem by (a) capturing multiple images of the same scene at varying exposure levels on a single sensor and combining them by time multiplexing to obtain a fused image that represents the highlight (bright) and shadow (dark) regions of an image in reasonable detail or by (b) using two sensors with a different sensitivity to light avoiding temporal disturbances.

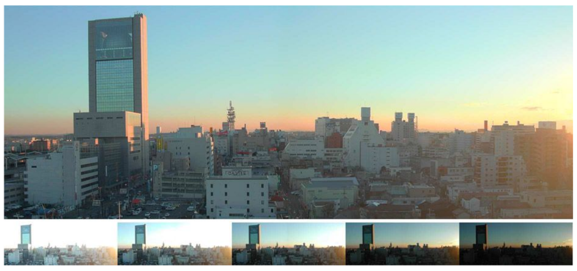


Figure 136: Fusing images with different exposure.

### 8.1.2 Autofocus (AF)

*Active* AF systems measure distance to the subject independently of the optical system, and subsequently adjust the optical system for correct focus. There are various ways to measure distance, including ultrasonic sound waves (e.g. some Polaroid cameras) and infrared light (early DSC & video cameras).

Passive AF systems determine correct focus by performing passive analysis of the image that is entering the optical system. They generally do not direct any energy, such as ultrasonic sound or infrared light waves, toward the subject. However, an autofocus assist beam of usually infrared light is required when there is not enough light to take passive measurements resulting in a hybrid system. Passive autofocusing can be achieved by phase detection (SLR) or contrast measurement (DSC, see below).

Active systems will typically not focus through windows, since sound waves and infrared light are reflected by the glass. With passive systems this will generally not be a problem, unless the window is stained. Accuracy of active AF systems is often considerably less than that of passive systems and therefore problematic when the DoF is small. Active systems may also fail to focus a subject that is very close to the camera since measurements get inaccurate. As a consequence, active systems are not used in microscopy.

Passive systems may not find focus when the contrast is low, notably on large single-colored surfaces (walls, blue sky, etc.) or in low-light conditions. Passive systems are dependent on a certain degree of illumination to the subject (whether natural or otherwise), while active systems may focus correctly even in total darkness when necessary. This is the motivation for the AF assist beam.

See `http://graphics.stanford.edu/courses/cs178/applets/` for nice applets on this and other topics.

**AF Phase Detection** The basic principle is like the split-image rangefinder focusing aid in a manual-focus SLR. This focusing aid consists of two shallow prisms, which angle your eye's view so it sees light rays coming from the two opposite edges of the lens. When the lens is correctly focused, these edge rays (by definition) must cross at the plane of the focusing screen; that means objects seen by the left edge of the lens and those seen by the right edge of the lens will line up with each other as seen through the split-image prisms. If the lens is incorrectly focused, the edge rays will cross either ahead of or behind the focusing screen; that means the rays from the left edge and right edge will be displaced relative to each other, and lines will appear "split" through the prisms.

The AF system works the same way, except that instead of the eye it uses a dedicated AF sensor consisting of two (CCD) arrays. Optics in the AF system work the same way as the split-image prisms, directing light from the left side of the lens to one CCD, and from the right side of the lens to the other CCD. The patterns of light and dark in the subject cause the individual elements of the CCD segments to put out different values, so that the total output of each CCD could be graphed as a wiggly, square-edged waveform corresponding to the light and dark patterns in the subject. Fig. 137(a) shows a ray diagram when the lens is in good focus, and (b) shows the intensity profile corresponding to this lens position. When the object is moved farther away, the rays from the upper and lower halves of the lens no longer intersect at the same locations, and the measured energy from the two halves of the lenses are out-of-phase (Figs. 137(c) and (d)) and requires the lens to move relative to the image plane to compensate for this defocus; in this case, towards the image plane.

Thus, the AF system's CPU compares the waveforms from the two CCDs to see whether or not they are "in phase" – if not, it can determine the amount and direction of the error based on the direction and displacement of the two waves relative to each other, and it uses this information to drive the AF motor to focus the lens. This is why phase detection is faster compared to contrast detection, since the latter requires iterative focusing and measuring stages to determine focus.

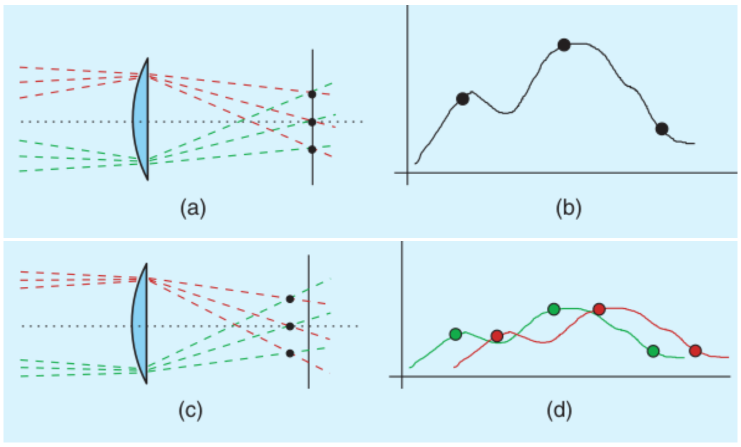


Figure 137: AF phase detection principle.

Fig. 138 illustrates how this is actually done in a camera since it is not entirely obvious how to get rays from the two halves of the lens.

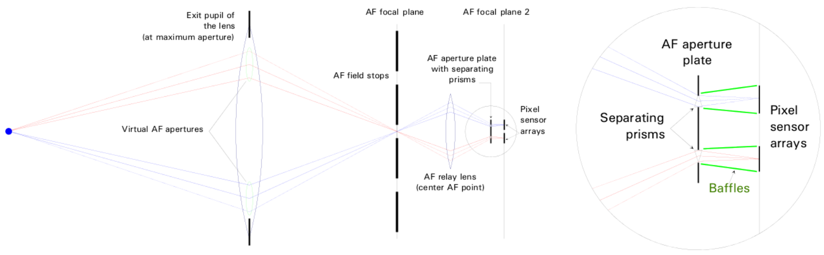


Figure 138: AF phase detection as used in SLR.

Because the image sensor is different from the focus sensor, there is a chance that they are not aligned and something considered focused by the focus sensor is not always focused on the image sensor. This is why phase-detect autofocus is more prone to front-/back-focusing issues (enthusiast/high-end cameras have a micro-adjust feature to address this issue).

**AF Contrast Detection** Contrast detection AF is achieved by measuring contrast (or similar values determining sharpness) within a sensor field, through the lens. The intensity difference between adjacent pixels of the sensor naturally increases with correct image focus. The optical system can thereby be adjusted until the maximum contrast is detected. In this method, AF does not involve actual distance measurement at all and is generally slower than phase detection systems, especially when operating under dim light. Furthermore, as the AF system cannnot calculate whether the subject is in front focus or back focus, iterative adjustment is required. As it does not use a separate sensor, however, contrast detection AF can be more flexible (as it is implemented in software) and potentially more accurate.

This, for a contrast detection AF system, the focusing process typically consists of two components: an image-based measure which indicates the sharpness of the image (i.e. the degree of focus), and a search algorithm which yields an image with the highest sharpness value. Depending on the target hardware system, the efficiency of the search strategies is determined by the number of sharpness evaluations (i.e. the number of images being taken and evaluated), the computational cost of each sharpness evaluation, and the number of stops and directional changes of the focusing adjustment system (i.e. cost of physical lens movement). The following search algorithms have been proposed:

- Global search: all possible focus positions are visited and the sharpness evaluated.
- Binary / Fibonacci search: A divide and conquer algorithm always breaks down the problem into two subproblems by partitioning the focus range into sets, two equally sized sets for binary search and two sets following the golden section rule for Fibonacci search.
- Hill Climbing: the maximum is searched by going into the direction of ascending values with some larger step-size, once the values start descending, search direction is reversed and small step size is used.
- Rule-based search: depending on the value of the gradient, the focus range is partitioned into four different types of areas where the number of analysed focus position is proportional to the gradient, also descending values are recorded and are used to steer the search process. A number of rules defines how to proceed under which conditions.
- Function fitting: the position of highest sharpness is predicted from some arbitrary measurement points by fitting a function of known shape or by using a neural network.

Let $I$ denote a set of digital images which are sorted in a way that they come from a defocused state to an intermediate focus and finally to an in-focus state. Subsequently, images get de-focused again. An autofocus function is a map $f : I \to \mathbb{R}$ with the characteristic that $f(i)$ is maximised as the image comes into focus. Further desired properties (which are eventually required to enable efficient search strategies, see below):

- The function should have only a single extremum, this avoids potential errors from local extrema. This means in other words that the function should be monotonically increasing towards its maximum and monotonically decreasing afterwards.
- The extremum should be attained when the system is in focus.
- The extremum should have a sharp peak.
- The function should react insensitively to other parameters that possibly change during the process like the mean brightness of the image.
- The function should be simple thus allowing for high execution speed.

While some focus search strategies do not require the underlying sharpness function to exhibit specific properties, the more efficient and intelligent schemes may significantly take advantage or even entirly rely on certain sharpness function properties to enable fast focusing. A global search is of the first type, since the sharpness function does not need to obey any specific property apart from attaining its maximum when the system is in focus. However, the number of evaluations is high when using this approach. Most of the techniques requiring a lower number of evaluations and lens movements rely on the assumption of a unimodal sharpness function, like binary and Fibonacci search or all variants of hill climbing. Less stringent sharpness function properties are necessary for rule-based search (i.e. a certain extent of continuity), as well as for function fitting by using functions of known shape or by using a neural network (in the latter case it is important for the sharpness function to exhibit the same shape independent of the underlying imagery). In any case, also the search strategies mentioned at last take advantage of unimodality of the sharpness function since the search will terminate faster and will be more accurate.

As usually speed is important, sharpness measures based on the application of initial integral transformations (like Fourier or wavelet transform) are usually not considered. Thus, we restrict the attention to spatial domain techniques, which can be divided into four main categories:

- **Functions based on differentiation**: As an image comes into focus edges become sharper and therefore the amount of high spatial frequencies increases. Image gradients are applied or the difference of the gray level intensity of pixels in the neighbourhood is calculated for computing focus measures. This category can be divided into methods that use the first derivative and methods that use the second derivative. Examples are given in equations (32), (33), (34), and (37).

- **Functions based on the histogram**: Histogram autofocus functions are based on the assumption that focused images have a greater number of grey levels than unfocused images. Defocused images are expected to be a single shade of gray, hence the number of bins in the histogram that contain occurrences increases as the image comes into focus. Examples are given in equations (35) and (36).

- **Functions based on statistical methods**: These methods calculate the variance or the standard deviation of the gray level intensities of an image. Also methods that use the autocorrelation functions can be found. This category can be divided into functions that are based on image contrast and those that are based on correlation measures. Examples are given in equations (41), (44) - (46).

- **Functions based on depth of peaks and valleys**: Local extrema of the intensity values and their distances are considered, based on the observation that peaks and valleys are better separated in focused images. Examples are given in equations (38) and (39).

Of course, there are also autofocus functions that combine several features of other autofocus functions. Examples are given in equations (42) and (46).

**Boddeke**: This method is based on applying a $(-1, 0, 1)$ filter mask along the horizontal $(x)$ axis of an image. The focus function is defined by squaring and adding all the filtered pixel values.

$$F_{Boddeke} = \sum_{x=1}^{X-1} \sum_{y=0}^{Y} [g(x+1, y) - g(x-1, y)]^2 \ , \tag{32}$$

where $X$ is the width of the image, $Y$ the height of the image and $g(x, y)$ the gray level intensity of pixel $(x, y)$.

**Brenner**: Brenner noted that as an image comes into focus, differences between a pixel and a pixel displaced for a certain amount increase:

$$F_{Brenner} = \sum_{x=0}^{X-n} \sum_{y=0}^{Y} [g(x, y) - g(x+n, y)]^2 \ , \tag{33}$$

where $n$ is a number specifying the amount of displacement.

**Laplace**: For analysing the high frequencies of the image it is convoluted with the Laplacian operator which is a second derivative operator:

$$L = \tfrac{1}{4} \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \ .$$

The Laplace focus measure is computed as follows:

$$F_{Laplace} = \sum_{x=n}^{X-n} \sum_{y=n}^{Y-n} |L(x, y)|, \tag{34}$$

where $L(x, y)$ is the convolution of $g(x, y)$ with the mask $L$ and $n$ defines the size of the Laplace operator, which means that $L(x, y)$ is computed as follows:

$$L(x, y) = \tfrac{1}{4} \cdot [g(x, y) \cdot 4 - g(x, y+n) - g(x-n, y) - g(x, y-n) - g(x+n, y)] \quad .$$

**Mendelsohn and Mayall's Histogram Method**: This method calculates the weighted sum of pixels in the histogram bins that are above a given threshold $T$ and is computed as follows:

$$F_{MenMay} = \sum_{x=0}^{X} \sum_{y=0}^{Y} \begin{cases} g(x, y) \cdot H_{g(x,y)}, & g(x, y) > T \\ \\ 0, & \text{else} \end{cases} \ , \tag{35}$$

where $H_{g(x,y)}$ is the number of pixels with intensity $g(x, y)$.

**Range**: Range is the difference between the maximum gray level and the minimum gray level, as an image comes into focus, the histogram range increases:

$$F_{Range} = max(g|H_g > 0) - min(g|H_g > 0) \ , \tag{36}$$

where $H_g$ is the number of pixels with intensity $g$.

**Tenengrad**: The Tenengrad autofocus function uses the Sobel operator for the calculation that in turn uses the two convolution masks

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} .$$

$$F_{Tenengrad} = \sum_{x=n}^{X-n} \sum_{y=n}^{Y-n} T(x,y) \ , \tag{37}$$

where $T(x,y) = S_x^2(x,y) + S_y^2(x,y)$ and $S_x(x,y)$ and $S_y(x,y)$ are the convolutions of the image with the Sobel operators $S_x$ and $S_y$. Again, $n$ determines the size of the operator.

**Thresholded Content**: This method adds the pixel values that are above a certain threshold $T$:

$$F_{Th\_Cont} = \sum_{x=0}^{X} \sum_{y=0}^{Y} \begin{cases} g(x,y), & g(x,y) \geq T \\ 0, & \text{else} \end{cases} . \tag{38}$$

**Thresholded Pixelcount**: This function computes the number of pixels below (above) a certain threshold:

$$F_{Th\_Pixelcount} = \sum_{x=0}^{X} \sum_{y=0}^{Y} s[g(x,y),T] \ , \tag{39}$$

with

$$s[g(x,y),T] = \begin{cases} 0, & g(x,y) \geq T \\ 1, & g(x,y) < T \end{cases} \tag{40}$$

**Variance and normalised Variance**: The Variance functions are based on image contrast, which is another feature that characterises sharpness since a well-focused image can be expected to show strong variation in gray levels.

$$F_{(Nor\_)Variance} = \frac{1}{XY\overline{g}} \sum_{x=0}^{X} \sum_{y=0}^{Y} [g(x,y) - \overline{g}]^2 \ , \tag{41}$$

where $\overline{g}$ is the mean of the gray level intensities of the image. For normalised variance, $1/\overline{g}$ is additionally used as a normalising factor to compensate for the differences in average image brightness among different images.

**Variance of Sobel**: The variance of the magnitude of the Sobel gradient is calculated.

$$F_{Var\_Sobel} = \sum_{x=n}^{X-n} \sum_{y=n}^{Y-n} (|S(x,y)| - \overline{S})^2 \ , \tag{42}$$

where $S(x,y) = \sqrt{S_x^2(x,y) + S_y^2(x,y)}$ and $\overline{S}$ is the mean of the absolute values of the Sobel gradient given by

$$\overline{S} = \frac{1}{(X-n)(Y-n)} \sum_{x=n}^{X-n} \sum_{y=n}^{Y-n} S(x,y) \ . \tag{43}$$

**Vollaths' Focusing Measures**: These measures are based on the autocorrelation function and the variance / standard deviation. We consider three variants:

$$F_{VollF4} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y} g(x,y) \cdot g(x+1,y) - \sum_{x=0}^{X-2} \sum_{y=0}^{Y} g(x,y) \cdot g(x+2,y) \ , \tag{44}$$

$$F_{VollF5} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y} g(x,y) \cdot g(x+1,y) - XY\overline{g}^2 \ , \tag{45}$$

$$F_{VollF11} = \frac{1}{XY(XY-1)}[XY \sum_{x=0}^{X-1} \sum_{y=0}^{Y} g(x,y) \cdot g(x+1,y) - (\sum_{x=0}^{X} \sum_{y=0}^{Y} g(x,y))^2] \ . \tag{46}$$

Several functions depend on a threshold or can be used with specific parameters, their behaviour often is significantly influenced by these parameters, Fig. 139 for the Laplace function with $n = 1$ and $n = 10$, all examples computed from sequences of 40 hardness testing images with different focus.

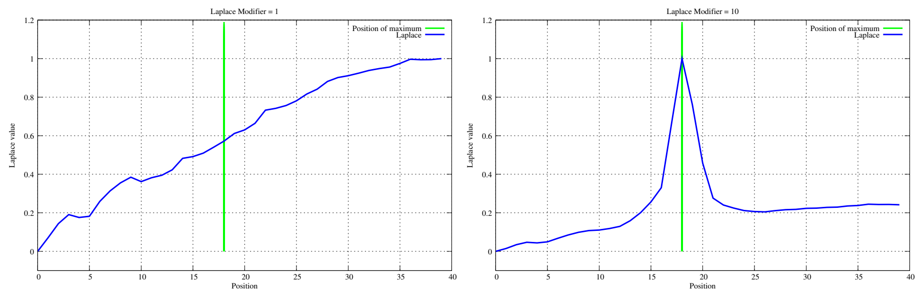


Figure 139: Laplace autofocus function after normalisation applied to a series of 40 images ($n = 1$, $n = 10$).

**Accuracy**: The most important criterion an autofocus function should fulfil is that the extremum should be attained when the image is in focus. This aspect is important for all focus search algorithms including full search of course. A way to score a function for this criteria is to use

$$F_{acc} = \frac{1}{1 + 0.25 \cdot (max_{found} - max_{true})^2} \ ,$$

where $max_{true}$ is the position of the sharp image in the image stack and $max_{found}$ is the position of the image in the image series that the autofocus function has computed. $F_{acc} = 1$, when the autofocus function has computed the right position. The higher the difference between $max_{true}$ and $max_{found}$ is, the more $F_{acc}$ goes towards 0. Fig. 140 shows an example with high and poor accuracy.

**Monotonicity**: For all focus search algorithms relying on a unimodal sharpness function (like hill-climbing etc.), the function should be monotonically increasing towards its maximum and monotonically decreasing afterwards. For that

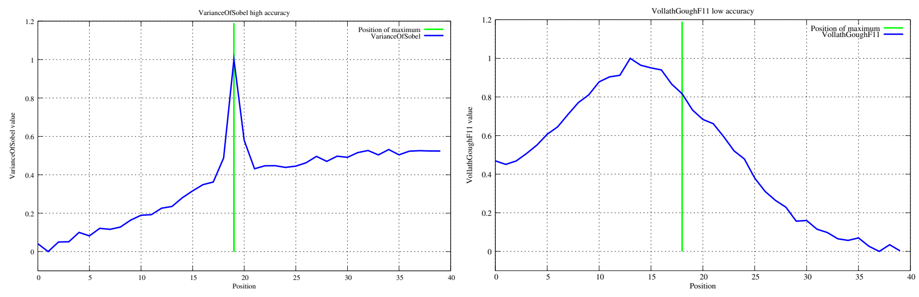


Figure 140: Accuracy of autofocus function: $F_{Var\_Sobel}$, $F_{acc} = 1$, vs. $F_{VollF11}$, $F_{acc} = 0.138$.

a function $F_{mon}$ has been used that calculates the differences of all $F(i)$ and $F(i+1)$ within the image series, where $F(i)$ denotes an autofocus functions' value of the image on position $i$. $F_{mon} = 1$, when the autofocus function is monotonically increasing towards its maximum and monotonically decreasing afterwards. The more often the monotonicity is disturbed, the more $F_{mon}$ goes towards 0. Therefore initially $F_{mon} = 1$, each time the monotonicity is disturbed, 0.075 is subtracted. When the value becomes negative $F_{mon} = 0$ and the algorithm stops. It should be noted that autofocus functions which produce more than a single extremum are scored low by $F_{mon}$. Fig. 141 shows an example with high and poor monotonicity.

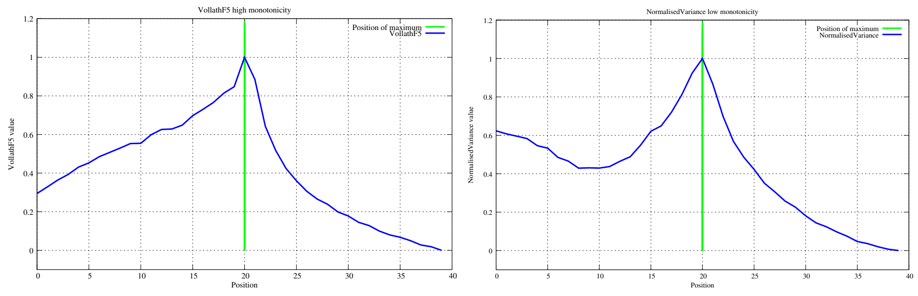


Figure 141: Monotonicity of autofocus function: $F_{VollF5}$, $F_{mon} = 1$ vs. $F_{Nor\_Variance}$, $F_{mon} = 0.325$.

**Peak Sharpness**: The sharpness of the peak is another criterion for selecting a good focus measure. A sharp peak makes algorithms possible that do a coarse search for the peak within the calculated values and come in a finer state when the values change more significantly. Especially two-step search and rule-based search may significantly benefit from distinct peak sharpness. To accomplish an assessment for that criterion a function $F_{sharp}$ has been developed that counts the values of an autofocus function that are above a focus level of 0.3. Few values are expected to be above that threshold if the autofocus function has a sharp peak, therefore $F_{sharp} = 1$, in case less than 20 percent of the values are above 0.3. The more values are higher than 0.3, the more $F_{sharp}$ goes towards 0, $F_{sharp} = 0$, as soon as more than 50 percent of the values are above 0.3. Fig. 142 shows an example with high and poor peak sharpness.

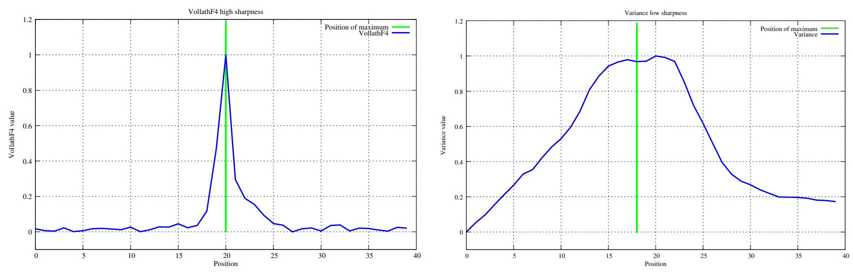


Figure 142: Peak Sharpness of autofocus function: $F_{VollF4}$, $F_{sharp} = 1$ vs. $F_{Variance}$, $F_{sharp} = 0$.

Regarding focus on white or other uniform areas, neither passive method will focus well if there isn't a contrast change in the image ... a solid white wall (or white portions of your test sheet) or concrete floor or blue sky with no clouds ... since the actual measurement is not about distance or intensity of light but rather contrast within the image. This is why some assist beam systems use a red grid pattern to help AF in low-contrast situations.

## 8.2 Colour Imaging Pipeline

The colour imaging pipeline operates on the acquired image, in most cameras based on three differently populated colour planes. The first processing block in the pipeline depicted in Fig. 143, "camera correction", is actually a collection of blocks as detailed in Fig. 144. The processing blocks required for a specific camera vary depending upon the hardware and the user expectations. Lower cost hardware typically leaves more artifacts in the raw image to be corrected, but the user expectations are often lower as well, so the choice of correction blocks used with a particular camera is the result of a number of system engineering and budget decisions. Few, if any, cameras use all of the processing blocks shown in Fig. 144.

n In some cases, users save images to a raw capture format and use sophisticated desktop software for processing, enabling a degree of control over the processing chain not usually exercised by the casual user. While these blocks are presented in a specific order in this discussion, the ordering chosen for a specific camera is dependent upon the causes of the artifacts needing correction and interactions between the effects. Usually, the preferred order of correction blocks is roughly the inverse of the order in which the artifacts are caused. Each of these correction blocks is complicated by the artifacts that have not yet been corrected. For example, if a dark correction is computed before defect concealment is completed, care should be taken to avoid using defective pixels in calculation of statistics for dark correction.

### 8.2.1 Channel Matching

The first correction discussed here is to match the response of multiple outputs or analog signal processing chains, such as with a dual output sensor. Because

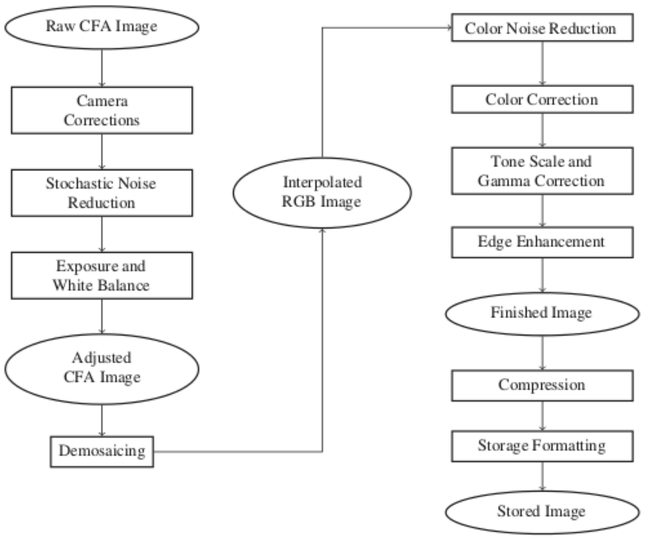


Figure 143: Color Imaging Pipeline: Detailed View.

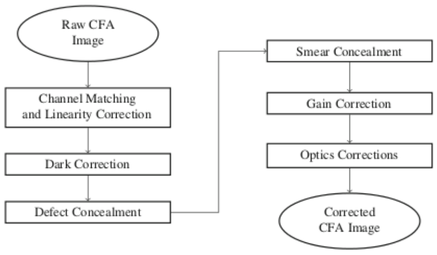


Figure 144: The stages of camera correction.

the artifacts due to channel mismatch are highly structured, usually a seam in the middle of the image or a periodic column pattern, the responses for the multiple outputs must match very closely. The most common form of this correction is to adaptively compute a dark offset correction for each output that will bring similar pixels from each output to match a common value, using reference dark pixels. The key to successful matching of multiple output channels is to take advantage of the knowledge of which image pixels came from which output.

### 8.2.2 Dark Correction

Dark correction is always necessary, since the analog output from the image sensor is rarely precisely “zero” for a zero light condition. Even with the lens cap on, a dark current signal is recorded, which is due to thermally generated electrons in the sensor substrate. To account for this, two strategies are used; place an opaque mask along the edges of the sensor to give an estimate of intensity due to dark current alone (this value can be corrupted by noise in the dark pixels, so some smoothing may be used to reduce the dark floor estimation error), or capture a dark image for the given exposure time (a second image is taken immediately after capturing the scene image with no exposure). In the first case, the mean dark current is subtracted from the entire image (this only works well in case of uniform dark floor), and in the second, the dark image itself is subtracted from the captured data.

In some cases, the dark floor is modeled using data from multiple dark captures. By averaging multiple dark captures, the impact of temporal noise on the dark floor estimate is minimized. This technique is still affected by changes in sensor tem- perature and integration time. Astronomical and other scientific applications, especially ones using a temperature-controlled sensor, routinely use this technique, made easier by the controlled temperature.

### 8.2.3 Defect Concealment

Sensor defects are somewhat problematic, since they indicate lost data that simply was not sensed. Algorithms for treating defects interpolate the missing data. The most common defects are isolated single pixel defects. Concealment of isolated pixels is usually done with a linear interpolation from the nearest adjacent pixels of the same color sensitivity (see *Demosaicing* section for interpolation techniques).

There are two way of treating these defects: First, by applying an impulse noise filter which tends to (inappropriately) filter out high-contrast details such as stars, lights, or specular reflections. Second, by maintaining a map of defective pixels depending upon a map from the sensor or camera manufacturer.

However, bright pixel defects caused by cosmic ray damage must be concealed without depending upon a preinstalled map, so a camera can implement a dark image capture and bright defect detection scan in firmware, usually done at startup. New defects found in the dark image are added to the defect map. Because cosmic ray damage tends to produce bright points rather than marginal defects, detecting these defects is relatively easy.

Sensor column defects or other more clustered defects caused e.g. by dirt on the cover glass of the sensor are much more difficult to correct as the latter also vary in size depnding on the focal length of the lens.

### 8.2.4 Smear Correction

Interline smear is a challenging artifact to correct or conceal because the artifacts vary with scene content. It is manifested as an offset added to some of the columns in the captured image. Since the added signal will usually vary from column to column, the effect will vary with the original scene content.

If a small amount of charge is added to pixels that are well below saturation, the artifact is manifested as a column that is brighter and lower in contrast than normal. If the sum of scene charge and smear charge saturates the pixels in the column, then the column looks like a bright defective column. Smear usually affects several adjacent columns, so saturated columns become difficult to conceal well.

Concealment approaches start with the use of dark rows or overclocked rows to estimate the smear signal that should be subtracted from each column. For example, one may subtract a smear signal from each column and apply a gain adjustment after the subtraction. The gain adjustment prevents bringing saturated columns down below the maximum code value, but adds gain variations to each column. In general, high quality smear correction is very difficult to achieve.

### 8.2.5 Gain Nonuniformity Correction

The correction of gain non-uniformity (caused by lens effects like vignetting or sensor characteristics) is esentially a multiplication of each pixel with a gain map. Early implementations, with very limited memory for storing gain corrections, used simple separable polynomials. Later implementations stored small images, with a gain value for each color channel for small tiles of the image. These maps were often created to make the sensor response to a uniform illumination completely flat, which left taking lens effects and interactions uncompensated.

With increasing adoption of CMOS sensors and evolution to smaller pixels, gain corrections now usually include lens interactions. For a camera with a fixed lens, these are relatively simple. For cameras with interchangeable lenses, this creates new overhead to combine a sensor gain map with a lens interaction gain map.

When lens effects get too severe, gain correction is usually limited to minimize noise amplification. This results as yet another system optimization, trading off darkness versus noisiness in the corners.

### 8.2.6 Optics Corrections

In addition to vignetting, geometric distorion, chromatic aberrations (longitudinal and lateral), and spatially varying reaction to an impulse light source. Geometric distortion is corrected by warping the image to invert the change in magnification, the extent of distortion is usually determined with calibration patterns like checkerboard images. Lateral chromatic aberration is corrected similarly by applying the procedure to colour bands separately. Longitudinal chromatic aberration (different color channels are focused at different distances from the lens) are treated by applying a sharpening filter to the affected colour bands. Because distortion correction may spatially resample the colour channels individually, it is often included in the processing chain after demosaicing. Convolution with a spatially varying kernel is used to compensate for spatially varying reaction to an impulse light source.

### 8.2.7 Stochastic Noise Reduction

All noise reduction operations seek to preserve as much scene information as possible while smoothing noise. To achieve this efficiently, it is important to use relatively simple models to discriminate between scene information and noise information.

In the stochastic noise reduction block, grayscale techniques for noise reduction are usually applied to each color channel individually, while after demosaicing, inter-colourband correlation may be exploited to distinguish noise from structural scene information ("colour noise reduction").

The first technique applied is range based filtering: This noise reduction is based on smoothing small intensity changes and retaining large ones. Thus, textures and edges with low contrast tend to get over-smoothed. The second artifact is the tendency to switch from smoothing to preservation when modulation gets larger. This results in a very nonuniform appearance in textured fields or edges, with portions of the texture being smoothed and other portions being much sharper.

The second technique is based on the likelihood that impulses are noise which leads to use of impulse filtering noise reduction, usually using a standard center-weighted median filter. The characteristic artifact caused by impulse filtering is elimination of small details from the scene, especially specular reflections from eyes and small lights. When applying impulse filters to CFA data, the filtering is particularly vulnerable to creating colored highlights, if an impulse is filtered out of one or two color channel(s), but left in the remaining channel(s).

### 8.2.8 Exposure and White Balance Correction

The HVS has the ability to map "white" colours to the sensation of white, even though an object has different radiance when it is illuminated with different light sources. In other words, a sheet of white paper under fluorescent lighting or under incandescent lighting or even under natural daylight appears to be white, although the actual irradiated energy produces different colors for different illuminations. This phenomenon is called *color constancy.*

DSC and SLR need to be taught how to map white under the capture illuminant to white under the viewing illuminant (and other colours accordingly). White balance adjustment is accomplished by multiplying pixels in each color channel by a different gain factor that compensates for a non-neutral camera response and illuminant imbalance. Application of the gain factors to the CFA data before demosaicing may be preferred, since some demosaicing algorithms may presume equal responses for the different color channels.

The camera¿s response to typical illuminants, such as daylight, incandescent, and fluorescent, is easily stored in the camera. In some circumstances, the capture illuminant is known (or can be determined). For example this is the case for flash usage or for user controlled illuminant selection on the camera. Another option to determine illuminant is to consider several possible illuminant classes and estimate the probability of each illuminant being the actual scene illuminant based on the colour characteristics.

In most cases, it is desirable to perform automated white balance, i.e. without knowledge about the capture illuminant. In this case, appropriate gain factors need to be determined to correct for illumination imbalance. Current cameras approach this estimation problem with different algorithms having different responses to scene content and illuminants. Camera manufacturers usually have somewhat different preferences, for example, biasing white balance to render images warmer or cooler, as well as different approaches to estimating the scene illuminant.

The best way to do white balance is to take a picture of a neutral object (white or gray) and deduce the weight of each channel. If the object is recorded as $R_w, G_w, B_w$, use weights $1/R_w, 1/G_w, 1/B_w$ for the three colour channels.

One means of performing auto white balance is to assume that a white patch must induce maximal camera responses in the three channels. The underlying theory is that highlights are specular reflections that are the colour of the illuminant. Thus, the white-balanced image has signals given by $R/R_{max}, G/G_{max}, B/B_{max}$ . However, the maximum in the three channels is very often a poor estimate of the illuminant and it does not work for scenes that have no truly specular highlights.

Most automatic white balance and exposure algorithms are based on some extension of the gray world model: Assume all colors in an image will average out to gray, $R = G = B$. Using this approach, the channels are scaled based on the deviation of the image average from gray. In this scheme, the white-balanced image has signals given by $k_r * R, G, k_b * B$, where $k_r = G_{mean}/R_{mean}$ and $k_b = G_{mean}/B_{mean}$.

However, the actual gray world model assumes that images of many different scenes will average out to 18% gray (a midtone gray). Unfortunately, this says very little about a specific image, but the algorithm must work well for individual images. Therefore, most extensions of the gray world model try to discount large areas of single colors, to avoid having the balance driven one way or another by red buildings, blue skies, or green foliage.

### 8.2.9 Demosaicing

Demosaicing is the process of generating three equally populated colourbands with full resolution from the image captured using a CFA technique. For this purpose, artificial data needs to be generated since all three colourbands are avaiable in subsampled form, i.e. the green channel has 50% and the red and blue channels have 25% of pixels populated, respectively. The technique used to generate these missing data is called *interpolation* – apart from demosiacing, interpolation is used in image resizing/scaling, defect concealment / correction (image impairment), superresolution, and many other techniques. Due to its importance, we first shed some light on basic principles of interpolation.

**Interpolation** Classical interpolation is the process to compute an interpolated value $g(x)$ at some (perhaps non-integer) coordinate $x$ as a linear combination of the samples $g_k$ evaluated at integer coordinates $k$, the weights being given by the values of the function $f(x-k)$:

$$g(x) = \sum_{k \in Z} g_k f(x-k) \ .$$

$f(x)$ must vanish for all integer arguments except at the origin, where it must be 1 (i.e. “interpolation property”). The summation is performed over all integer coordinates, however, in practice the number of known (or used) samples is always finite. A large variety of different “interpolation kernels” $f(x)$ is used, having different properties with respect to resulting quality of the interpolated data, execution speed of the computation, memeory requirement etc.

The *nearest neighbour* kernel is the simplest of all: $f_{NN}(x) = 1$ for $-0.5 \leq x < 0.5$, and $f_{NN}(x) = 0$ if $x < -0.5$ and $x \geq 0.5$. For any coordinate $x$ where it is desired to compute the value of the interpolated function $g$, there is only one sample $g_k$ that contributes. Thus, the main interest of this appraoch is its simplicity, the price to pay is a very low quality.

*Linear interpolation* still offers very low complexity but improves quality as compared to nearest neighbour interpolation considerably: $f_{LIN}(x) = 1 - |x|$ for $|x| < 1$ and 0 otherwise ($|x| \geq 1$). How does this correspond to our usual notion of taking the sum and divide by two ? For example, consider two pixel values (6 and 10) next to each other and we want to compute the interpolated value right in the middle of them. Following our general formula, we result in:

$$g(0) = 10 f_{LIN}(-0.5) + 6 f_{LIN}(0.5) = 5 + 3 = 8 \ .$$

This is exactly the result we expect. In two dimensions (as visualised in Fig. 145 left), also called bilinear interpolation, its separable implementation requires four samples. Here, first columns are interpolated, followed by an interpolation of the lines.

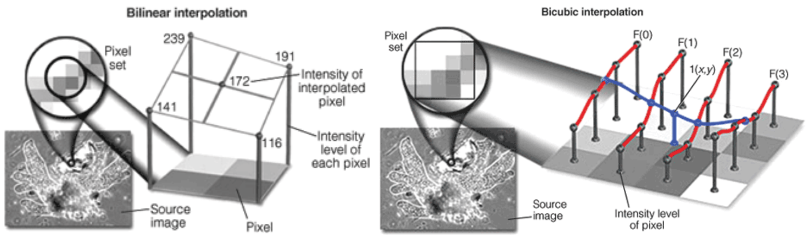


Figure 145: Bilinear and Bicubic interpolation

*Cubic interpolation* produces less blurring of edges and other distortion artifacts than bilinear interpolation, but is more computationally demanding. Polynomials of third degree are used as kernel functions such that more sample points can be considered. Bicubic interpolation involves fitting a series of cubic polynomials to the pixels contained in a $4 \times 4$ array of pixels surrounding the calculated address. First, four cubic polynomials are fitted to the control points in the y-direction (the choice of starting direction is arbitrary). Next, the fractional part of the calculated pixel's address in the y-direction is used to fit another cubic polynomial in the x-direction, based on the interpolated pixel values that lie on the curves. Substituting the fractional part of the calculated pixel's address in the x-direction into the resulting cubic polynomial then yields the interpolated pixel's brightness value. Such bicubic interpolation has found use in many commercial software packages such as Adobe Photoshop and many more.

The choice of polynomial used in the (bi)cubic interpolation algorithm can have a significant impact on the accuracy and visual quality of the interpolated image. In the following, we demonstrate how to derive a cubic interpolation kernel function $f_{CUB}(x) = f(x)$.

If the values of a function and its derivative are known at $x = 0$ and $x = 1$, then the function can be interpolated on the interval $[0, 1]$ using a third degree polynomial $f(x) = ax^3 + bx^2 + cx + d$, $f'(x) = 3ax^2 + 2bx + c$. The values of the polynomial and its derivative at $x = 0$ and $x = 1$ are given as $f(0) = d$, $f(1) = a + b + c + d$, $f'(0) = c$, and $f'(1) = 3a + 2b + c$. The four equations can be rearranged so that they deliver the required polynomials' coefficients $a = 2f(0) - 2f(1) + f'(0) + f'(1)$, $b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$, $c = f'(0)$, and $d = f(0)$.

However, in most cases (particularly in image processing), we do not know the derivative of the underlying (image intensity) function, but we simply want to interpolate between a list of pixels. Instead of setting the derivative to 0 at each point (which does not lead to smooth curves), we use the slope of a line between the previous and the next point as the derivative at a point (the resulting kernel is called "a Catmull-Rom spline"). Suppose we have the samples $g_0$, $g_1$, $g_2$, and $g_3$, at the positions $x = -1$, $x = 0$, $x = 1$, and $x = 2$ . Then we can assign the values of $f(0)$, $f(1)$, $f'(0)$ and $f'(1)$ using the formulas below to interpolate between $g_1$ and $g_2$: $f(0) = g_1$, $f(1) = g_2$, $f'(0) = \frac{g_2 - g_0}{2}$, and $f'(1) = \frac{g_3 - g_1}{2}$. Setting these values into the above formula for the polynomial coefficients we result in $a = -1/2g_0 + 3/2g_1 - 3/2g_2 + 1/2g_3$, $b = g_0 - 5/2g_1 + 2g_2 - 1/2g_3$, $c = -1/2g_0 + 1/2g_2$, and $d = g_1$, resulting in the corresponding polynom $f(g_0, g_1, g_2, g_3, x)$.

For bicubic interpolation, suppose we have the 16 samples (pixels) $g_{ij}$ with $i$ and $j$ going from 0 to 3 and with $g_{ij}$ located at $(i-1, j-1)$. Then we can interpolate the area $[0, 1]^2$ by first interpolating the four columns and then interpolating the results in the horizontal direction. The formula for the polynom becomes:

$$\begin{aligned} f(x, y) &= f(f(g_{0,0}, g_{0,1}, g_{0,2}, g_{0,3}, y), f(g_{1,0}, g_{1,1}, g_{1,2}, g_{1,3}, y), && (47) \\ &\quad f(g_{2,0}, g_{2,1}, g_{2,2}, g_{2,3}, y), f(g_{3,0}, g_{3,1}, g_{3,2}, g_{3,3}, y), x) \; . && (48) \end{aligned}$$

Alternatively, the formula can be derived if the function values of $f(x, y)$, $f_x(x, y)$, $f_y(x, y)$, and $f_{xy}(x, y)$ at the four corners $(0, 0)$, $(1, 0)$, $(0, 1)$, and $(1, 1)$ are known. The unknown coefficients $a_{ij}$ of the corresponding 2-D polynomial surface $f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$ can be computed by solving a system of 16 linear equations, similar to the procedure above for the one dimensional case.

As an example, we compute an interpolation polynomial (the green curve) for the four points on the red curve, both shown in Fig. 146. We have given the four points $g_0 = 2$, $g_1 = 4$, $g_2 = 2$, and $g_3 = 3$, at the positions $x = 1$, $x = 2$, $x = 3$, and $x = 4$ (note that the x-positions are different compared to those used in the derivation).

We compute the resulting polynomials' coefficients as $a = -1/2 * 2 + 3/2 * 4 - 3/2 * 2 + 1/2 * 3$, $b = 2 - 5/2 * 4 + 2 * 2 - 1/2 * 3$, $c = -1/2 * 2 + 1/2 * 2$, and $d = 4$, resulting in the polynom $f(x) = 7/2(x - 2)^3 - 11/2(x - 2)^2 + 4$ (x-2 replaces x due to the shift from $[0, 1]$).

Bicubic spline interpolation as demonstrated requires the solution of the linear system described above for each grid cell. A fixed kernel with similar properties is often used instead (as derived by Keys): $f_{KEYS}(x) = (a+2)|x|^3 - (a+3)|x|^2 + 1$

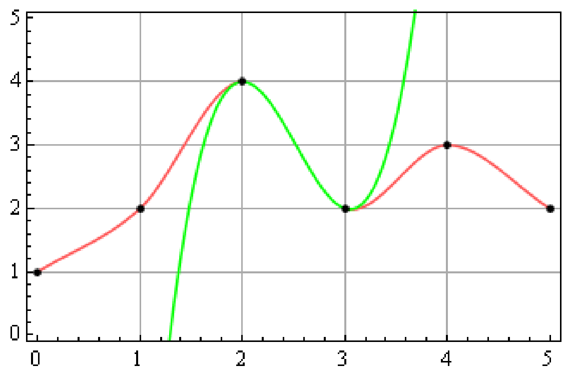


Figure 146: Example for cubic interpolation

for $0 \leq |x| < 1$, $f_{KEYS}(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a$ for $1 \leq |x| < 2$, and $f_{KEYS}(x) = 0$ for $|x| \geq 2$. Often, a fixed choice is $a = 0.5$.

Many more interploation kernels do exist, like the Lanczos kernel or the Sinc kernel, or various types of spline interpolation methods.

**Interpolating CFA generated Data** An important issue for these algorithms is computational cost and easy of hardware implementation. It has to be noted that many of the techniques described are covered by patents of the respective camera producers. Often, the actual technique used in a camera is not publicly known.

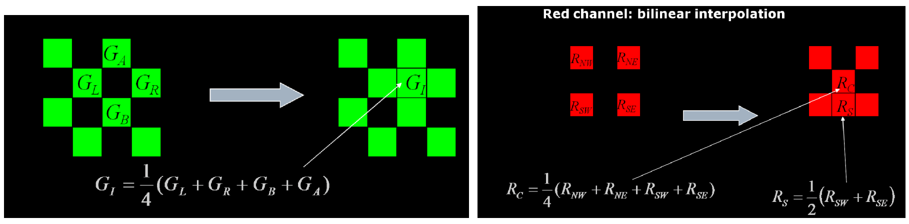

$G_A$
$G_L$
$G_R$
$G_B$
$G_I$
$G_I = \frac{1}{4}(G_L + G_R + G_B + G_A)$
Red channel: bilinear interpolation
$R_{NW}$
$R_{NE}$
$R_{SW}$
$R_{SE}$
$R_C$
$R_S$
$R_C = \frac{1}{4}(R_{NW} + R_{NE} + R_{SW} + R_{SE})$
$R_S = \frac{1}{2}(R_{SW} + R_{SE})$

Figure 147: Bilinear colorplane interpolation

The first and most obvious approach is to apply interpolation techniques to each colour plane independently. Nearest neighbour interpolation makes an arbitrary choice which pixel is selected for identical distance, Fig. 147 shows the scheme used for the G and R,B colourplanes, respectively. The examples in Fig. 148 illustrate that significant colour artifacts arise when applying this strategy. It has to be noted that also when applying more advanced interpolation (like bicubic techniques), those effects cannot be reduced significantly.

The effect displayed in colled "Colour Moire effect (or colour fringes or zipper effect)" and is caused by misinterpreting luminance detail as colour information. Caused by the poor interpolation results of individual clourplane interpolation, sharp luminance transitions cause a sharp transition in the colour planes at different spatial locations, i.e. the colour planes do not react in a synchronized manner to sharp edges. An example of this effect and the situation causing the

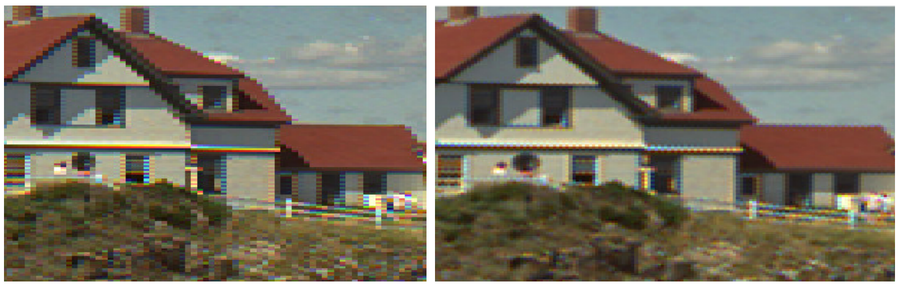


Figure 148: Examples for color plane interpolation: nearest neighbour vs. bilinear

effect is shown in Fig. 149.

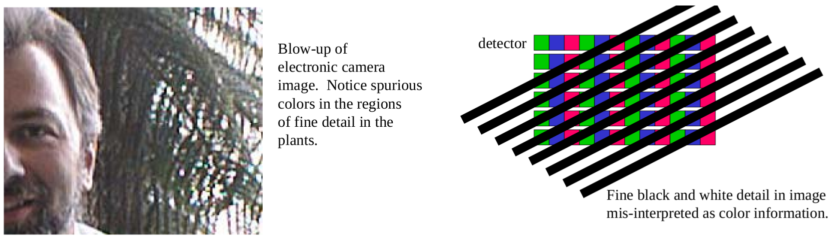

Blow-up of electronic camera image. Notice spurious colors in the regions of fine detail in the plants.
detector
Fine black and white detail in image mis-interpreted as color information.

Figure 149: Color Moire artefact

Fig. 150 illustrates what exactly happens at a sharp luminance transition. As a consequence, it is imperative to incorporate the inter-colourband correlations into the demosaicing process. A significant number of corresponding approaches have been suggested throughout the last 2 decades, the tendency is to increase complexity resulting in steadily increasing quality in this field.

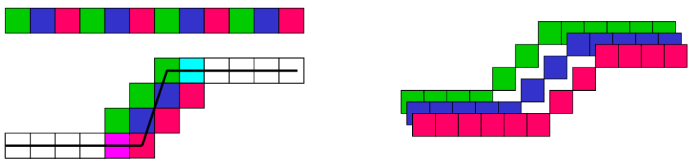


Figure 150: Principle of color sampling errors

The first approach employs a *median filter* to colourplane differences. The idea is clear: since the colourplanes are out of synchronisation, a difference signal contains isolated maximas in areas where colour fringe occurs (see Fig. 151 for the R-G signal). Therefore, as illustrated in the figure, a median filter is applied to colour difference signals, the results of which are used with original measurements to compute all the RGB values in each pixel. This is possible as we have one value and two differences for each pixel.

Fig. 152 shows an example of a filteres difference signal (R-G signal) and a comparison of the colourplane independent bilinear interpolation and the median filtering approach, where the latter shows clearly reduced colour artefacts.

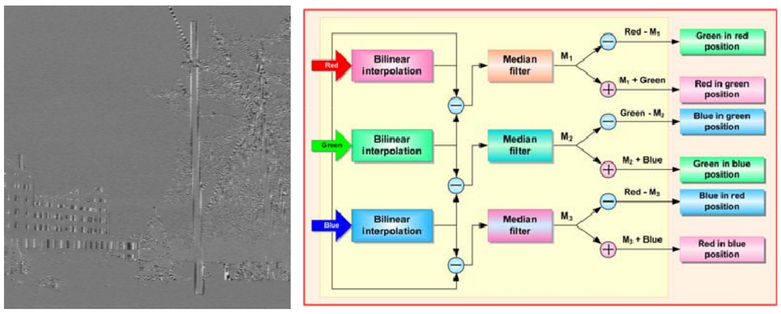


Figure 151: Median filtering approach

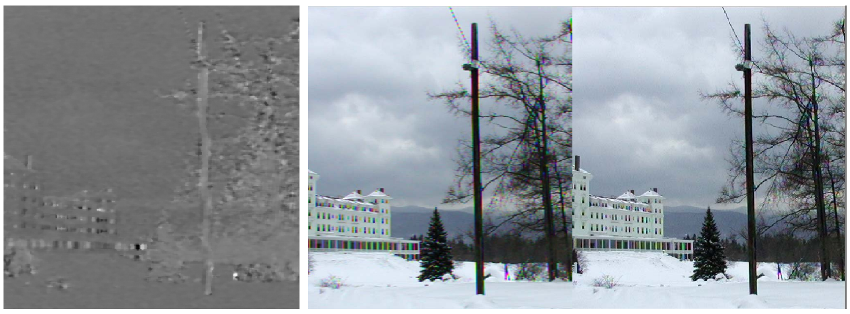


Figure 152: Median filtering result

The median filtering approach is a first approach to take into account the strong spectral correlation between color com- ponents at each pixel. Two main hypotheses are proposed in the literature in this context. The first one assumes a *color ratio constancy* and the second one is based on *color difference constancy* (where median filtering obviously relies on the latter). Interpolation based on color hue constancy follows the first idea (where *hue* is understood as the ratio between chrominance and luminance, i.e. R/G or B/G when the G plane is identified with luminance as it is often done), which exibits problems in case the denominator G takes low values. This happens for instance when saturated red and/or blue components lead to comparatively low values of green, making the ratios R/G and B/G very sensitive to red and/or blue small variations.

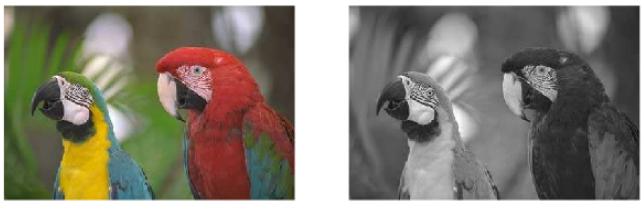


Figure 153: Original image and G plane

Fig. 153 is a natural image example which is highly saturated in red and Fig. 154 shows the images where each pixel value is, respectively, the component ratio R/G and difference R ¿ G. It can be noticed that these two images actually carry out less high-frequency information than the green component plane.

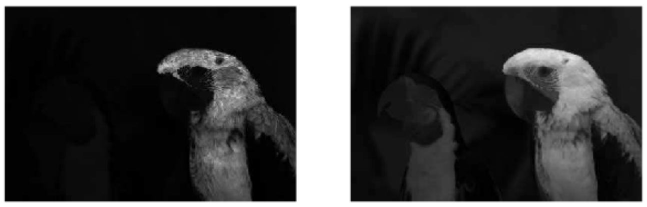


Figure 154: R/G and R-G planes

A Sobel filter is then applied to these two images, so as to highlight the high-frequency information location as shown in Fig. 155. In the right-hand parrot plumage area where red is saturated, the component ratio plane contains more high-frequency information than the component difference plane, which makes it more artifact-prone when demosaiced by interpolation. Moreo- ver, high color ratio values may yield to estimated component levels beyond the data bounds, which is undesirable for the demosaicing result quality.

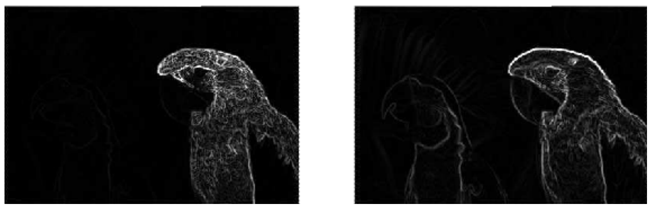


Figure 155: Sobel filter output of R/G and R-G planes

Constant hue transition interpolation first interpolates the G plane by some desired method (by-linear or edge-directed, see below). Using the assumption that hue is smoothly changing across an objects surface, the hue value is interpolated and the interploation for the chrominance values are derived from the interpolated hue values. To be more specific, the interpolated R hue (R/G ratio) and B hue (B/G ratio) are multiplied by the G value to determine the missing R and B values at a given pixel position.

For example, refering to the Bayer pattern in Fig. 156, the following formulas are used:

$$R_{44} = G_{44} \frac{\frac{R_{33}}{G_{33}} + \frac{R_{35}}{G_{35}} + \frac{R_{53}}{G_{53}} + \frac{R_{55}}{G_{55}}}{4}$$

$$B_{33} = G_{33} \frac{\frac{B_{22}}{G_{22}} + \frac{B_{24}}{G_{24}} + \frac{B_{42}}{G_{42}} + \frac{B_{44}}{G_{44}}}{4}$$

Note that the G values involved in these formulas are the result of the first interpolation stage. Fig. 156 illustrates how this concept can be used employing colourband differences instead of hue.

Nonadaptive demosaicing algorithms typically provide satisfactory results in smooth image regions, while they usually fail in textured regions and edges.

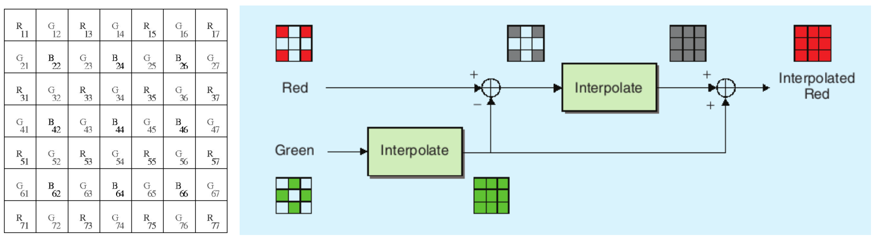
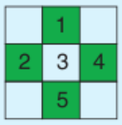

R11 G12 R13 G14 R15 G16 R17
G21 B22 G23 B24 G25 B26 G27
R31 G32 R33 G34 R35 G36 R37
G41 B42 G43 B44 G45 B46 G47
R51 G52 R53 G54 R55 G56 R57
G61 B62 G63 B64 G65 B66 G67
R71 G72 R73 G74 R75 G76 R77
Red
Green
Interpolate
Interpolate
Interpolated Red
+
−
+
+

Figure 156: Bayer CFA pattern and constant-difference-based interpolation

*Edge-directed interpolation* is an adaptive approach, where the area around each pixel is analysed to determine if a preferred interpolation direction exists. In practice, the interpolation direction is chosen to avoid interpolation across edges, instead interpolating along any edges in the image. Fig. 157 shows an example of applying this idea to a single colour band (thus it can also be applied to any grayscale image and is therefore also a generic adaptive interpolation approach). In practice, the gradients themselves and their difference should exceed some threshold. The idea can of course be combined also with bicubic or any other mode advance dtechnique.



```
1. Calculate horizontal gradient $\Delta H = |G2 - G4|$
2. Calculate vertical gradient $\Delta V = |G1 - G5|$
3. If $\Delta H > \Delta V$,
       $G3 = (G1 + G5)/2$
   Else if $\Delta H < \Delta V$,
       $G3 = (G2 + G4)/2$
   Else
       $G3 = (G1 + G5 + G2 + G4)/4$
```

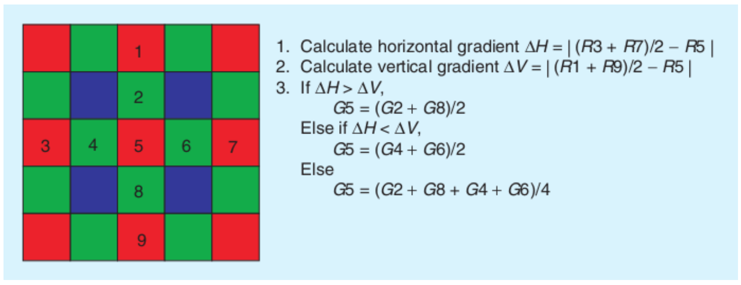
Figure 157: Edge-directed interpolation on a single colour plane.

In Fig. 158, this idea is extended to exploit inter-colourband correlation as well. Here, the R and B channels in a larger neighbourhood are used instead of the G channel to determine gradients, second-order derivatives are used. Once the luminance is determined, chrominance values are are interpolated from the differences bewteen the colour (R and B) and luminance (G) channels (again, ratios could be used as well). For example (notation of Fig. 158 is used),

$$R8 = \frac{(R5 - G5) + (R9 - G9)}{2} + G8 \text{ and } R4 = \frac{(R3 - G3) + (R5 - G5)}{2} + G4 \; .$$

and for the red value in a blue pixel the four differences NW, NE, SW, and SE are added, divided by four, and the corresponding G interpolation value is added.

*Adaptive colour plane interpolation* improves the approach by also using colour plane information to interpolate the green band, i.e. (notation of Fig. 158 is used)



```
1. Calculate horizontal gradient $\Delta H = |(R3 + R7)/2 - R5|$
2. Calculate vertical gradient $\Delta V = |(R1 + R9)/2 - R5|$
3. If $\Delta H > \Delta V$,
        $G5 = (G2 + G8)/2$
   Else if $\Delta H < \Delta V$,
        $G5 = (G4 + G6)/2$
   Else
        $G5 = (G2 + G8 + G4 + G6)/4$
```

Figure 158: Edge-directed interpolation involving all colour planes.

$$G5 = \frac{G2 + G8}{2} + \frac{2 * R5 - R1 - R9}{2} \text{ for } \delta V < \delta H \ ,$$

$$G5 = \frac{G4 + G6}{2} + \frac{2 * R5 - R3 - R7}{2} \text{ for } \delta V > \delta H \ \text{, and}$$

$$G5 = \frac{G2 + G4 + G6 + G8}{4} + \frac{4 * R5 - R1 - R3 - R7 - R9}{4} \text{ for } \delta V = \delta H \ .$$

Here, in fact second order colour gradients are used in the interpolation, in the original scheme, also $\delta V$ and $\delta H$ are more complicated. The colour channels are interpolated using a similar technique. A further refinement is to use more directions for computing gradient information.

The technique called *High Quality Linear Interpolation* as used in e.g. Matlab is very similar but uses different weights in its interploation scheme. 8 different cases are distinguished: 2 to determine the red and green values on a blue pixel, 2 to determine the blue and green values on a red pixel, and 4 to determine the red and blue values on a green pixel (2 for a green pixel in a "red row" and 2 in a "blue row"). Fig. 159 schows the corresponding 8 interpolation schemes (which need to be normalised before application).

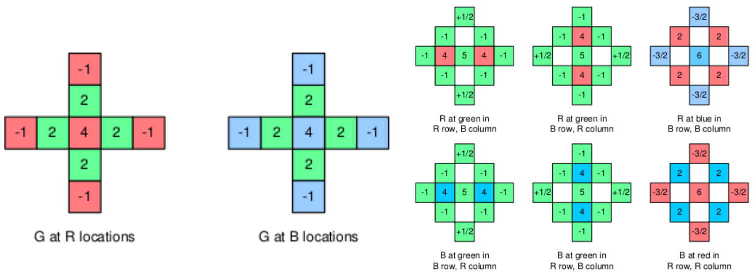

G at R locations
G at B locations
R at green in R row, B column
R at green in B row, R column
R at blue in B row, B column
B at green in B row, R column
B at green in R row, B column
B at red in R row, R column

Figure 159: 8 interpolation schemes for High Quality Linear Interpolation

Further demosaicing techniques include:

- *Pattern Recognition Interpolation*: This family of methods aims at identifying a template-based feature in each pixel neighborhood, in order to interpolate according to the locally encountered feature. The first step in his procedure is to find the average of the four neighboring green pixels, and classify the neighbours as either high (h) or low (b) in comparison to this average (see Fig. 160).

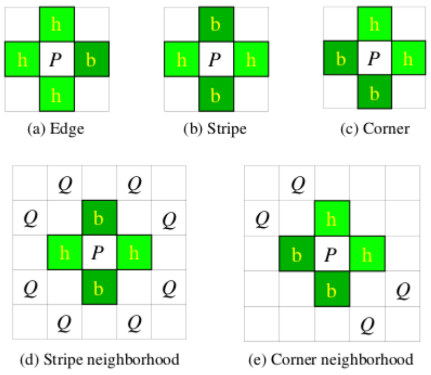


Figure 160: Patterns used to determine the central pixel interpolation

These values are sorted and denoted as $G_1, \ldots, G_4$, $M = \frac{G_2+G_3}{2}$. The green pixel $\hat{G}$ is then defined as an edge if three neighbor pixels share the same classification. If not, then the pixel can either be a part of a corner or a stripe. If two adjacent neighbour pixels have the same classification, then the pixel is a corner. If two opposite pixels have the same classification, then the pixel is a stripe. If an edge is detected, $\hat{G} = M$, for a stripe $\hat{G} = CLIP(M - (S - M))$ where $S$ is the average green level over the eight neighboring pixels labeled as Q in the figure. For a corner, $\hat{G} = CLIP(M - (S' - M))$ where where $S'$ is the average green level over the eight neighboring pixels labeled as Q in the figure. $CLIP$ limits the interpolated value to $[G_3, G_2]$. The other colour planes can be interpolated using any of the techniques described before.

- *Homogeneity-directed interpolation*: The RGB data is first interpolated horizontally and vertically, i.e., there are two candidates for each missing color sample. Both the horizontally and vertically interpolated images are transformed to the CIELAB space. In the CIELAB space, either the horizontally or the vertically interpolated pixel values are chosen based on the local homogeneity. The local homogeneity is measured by the total number of similar luminance and chrominance values of the pixels that are within a neighborhood of the pixel in question.

- *Vector-based interpolation*: In this approach, each pixel is considered as a vector in the three dimensional (R,G,B) space, and interpolation is designed to minimise the angle or the distance among neighbouring vectors. After an inital interpolation of missing samples, each pixel is transformed to spherical coordinates $(\rho, \Phi, \phi)$:

$$R = \rho \cos(\Phi) \sin(\phi)\ ,\ R = \rho \cos(\Phi) \cos(\phi)\ ,\ B = \rho \sin(\Phi)\ .$$

In the $(\rho, \Phi, \phi)$ space, a filtering operation like median filtering is applied to the angles only. This forces the chrominance components to be similar. Because $\rho$ is closely related to the luminance component, keeping it unchanged preserves the luminance discontinuities among neighboring pixels. After the filtering process, the image is transformed back to the (R, G, B) space.

### 8.2.10 Colour Noise Reduction

### 8.2.11 Colour Correction

### 8.2.12 Tone Scale and Gamma Correction

### 8.2.13 Edge Enhancement - Sharpening

### 8.2.14 Compression

This topic will be covered in-depth in the subsequent lecture "Multimedia Data Formats".