Grundlagen Bildverarbeitung

Univ.-Prof. Dr. Andreas Uhl

WS 2011/2012

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
- 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 ...

- IEEE International Conference on Image Processing (ICIP)
- IEEE International Conference on Accustic, Speech and Signal Processing (ICASSP)
- SPIE Symposium on Electronic 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.

[1] image processing $\neq$ imaging!

- 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 content of images.

  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 computer vision 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

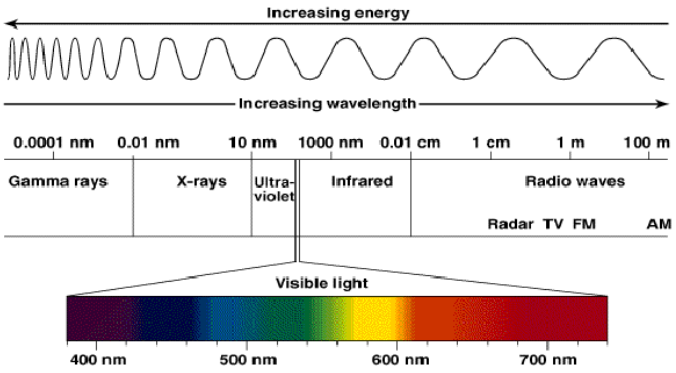


Figure 1: Electromagnetic spectrum

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 !

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 ?

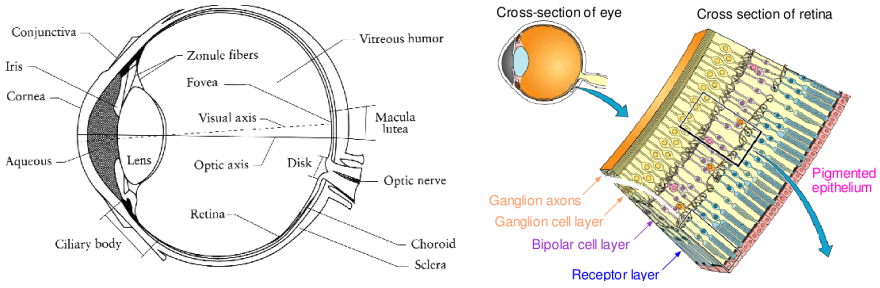


Figure 2: Human eye and retina.

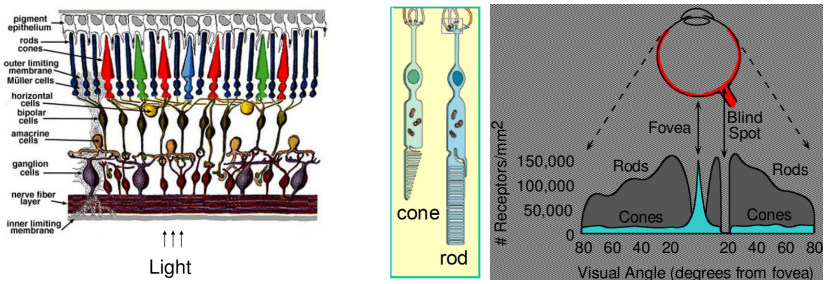


Figure 3: Color und luminance perception: Rods and cones.

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).

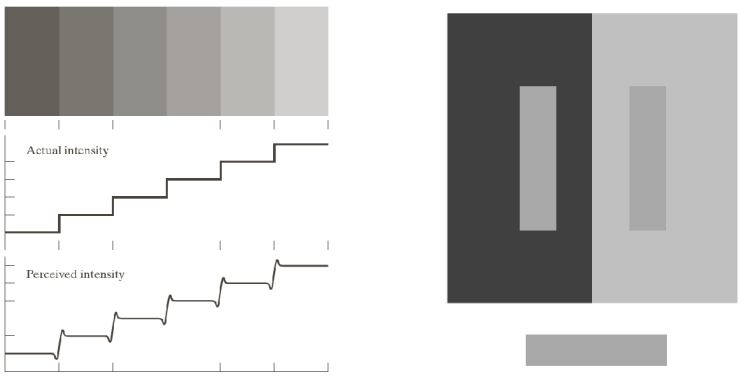


Figure 4: Perceived intensity depends on surroundings.

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.

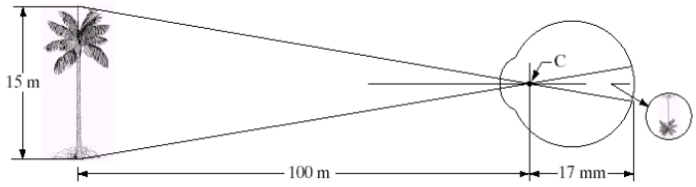


Figure 5: Image formation in the eye

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.

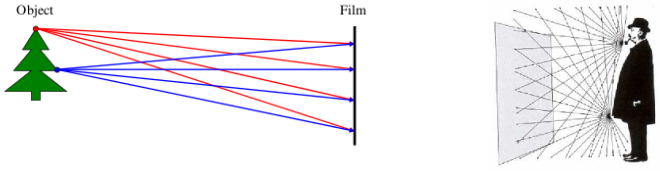


Figure 6: Using a plain sensor as a camera does not work.

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.

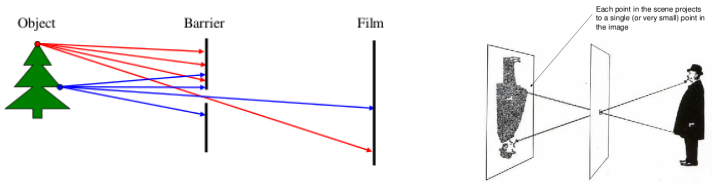


Figure 7: Using a barrier with hole improves the situation.

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.

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

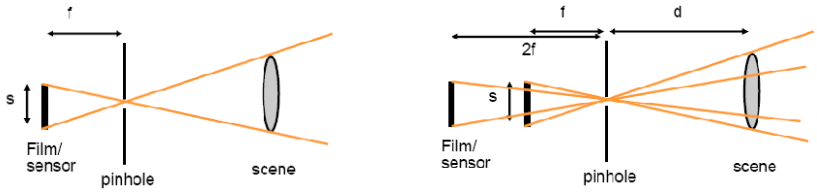


Figure 8: Influence of focal length on object size.

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).

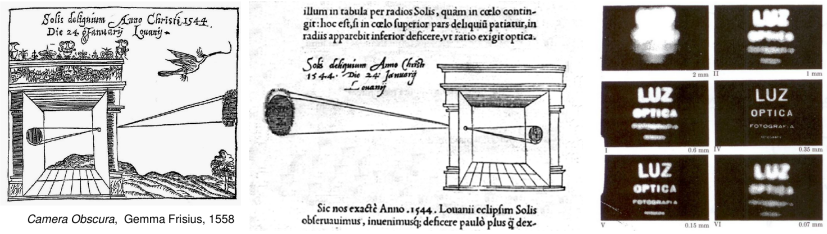


Figure 9: Historic pinhole cameras and tradeoff size of pinhole / sharpness.

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).

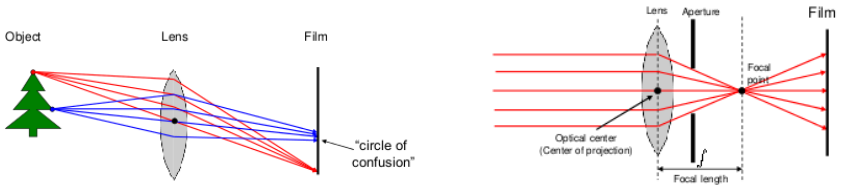


Figure 10: Introduction of the lens.

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

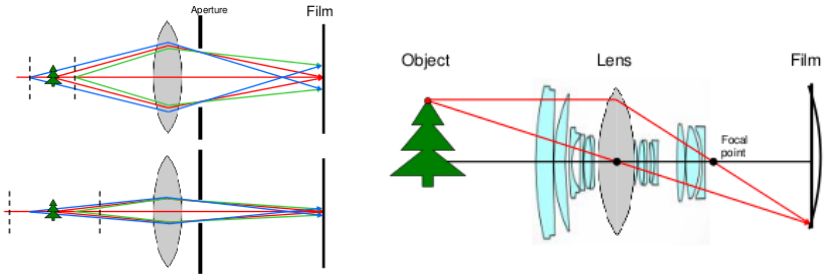


Figure 11: Effects of changing aperture and thick lenses.

is shown. We denote by y the object size and by y' the size of the object in the 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.

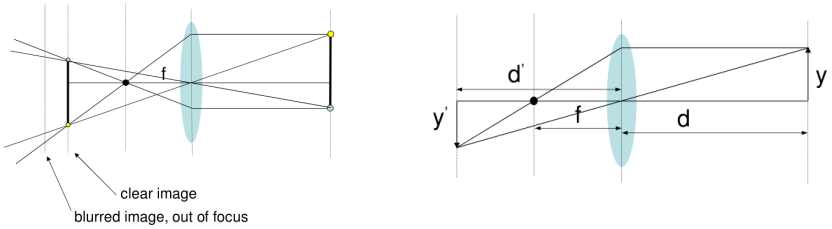


Figure 12: Thin lens scenario.

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

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

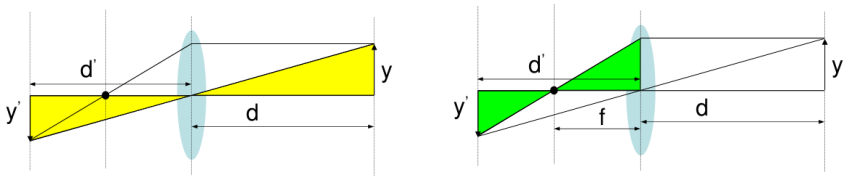


Figure 13: Deriving the thin lens formula.

Equating the right side of both equations we result in

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

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.

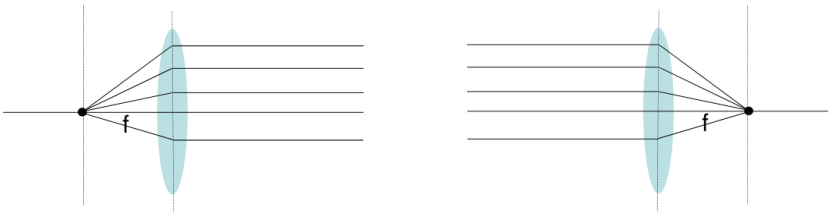


Figure 14: Extreme settings.

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).

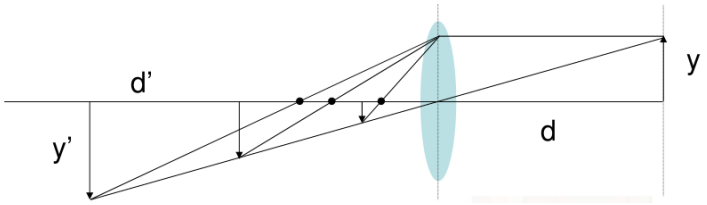


Figure 15: Effect of focal length on image size.

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.

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).

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

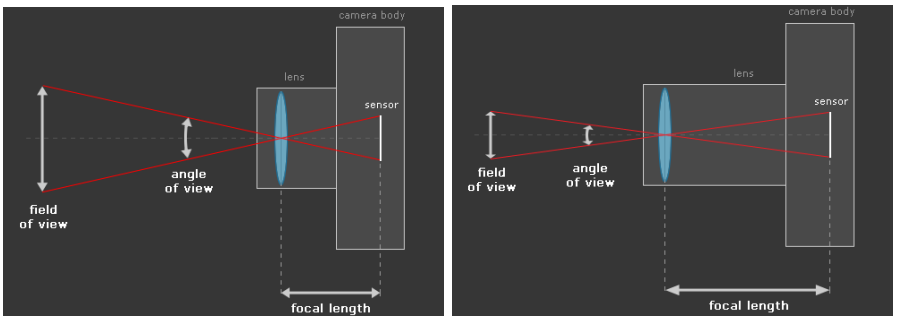


Figure 16: Effect of focal length on image content.

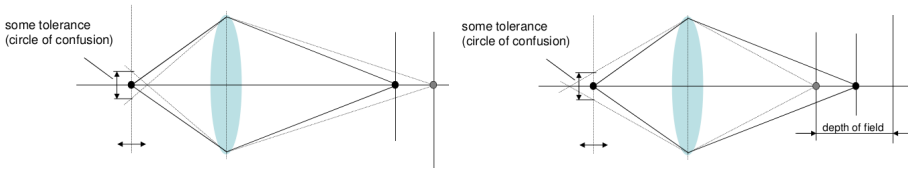


Figure 17: Depth of field and circle of confusion.

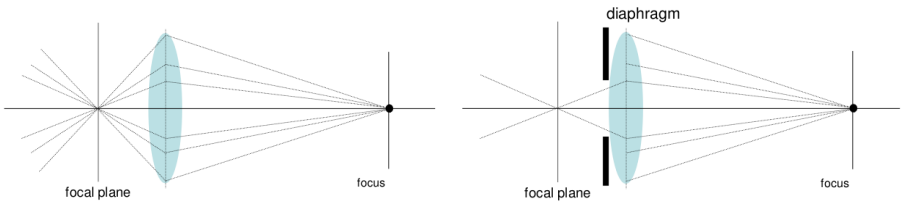


Figure 18: Effect of using an aperture (object in focus).

sharper (less blurry) image (see Fig. 19).

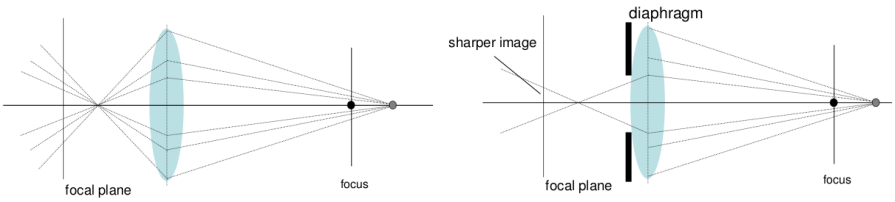


Figure 19: Effect of using an aperture (object non fucused).

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:

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

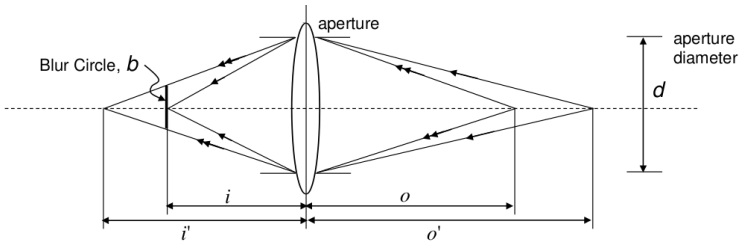


Figure 20: Computing the size of the circle of confusion.

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 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

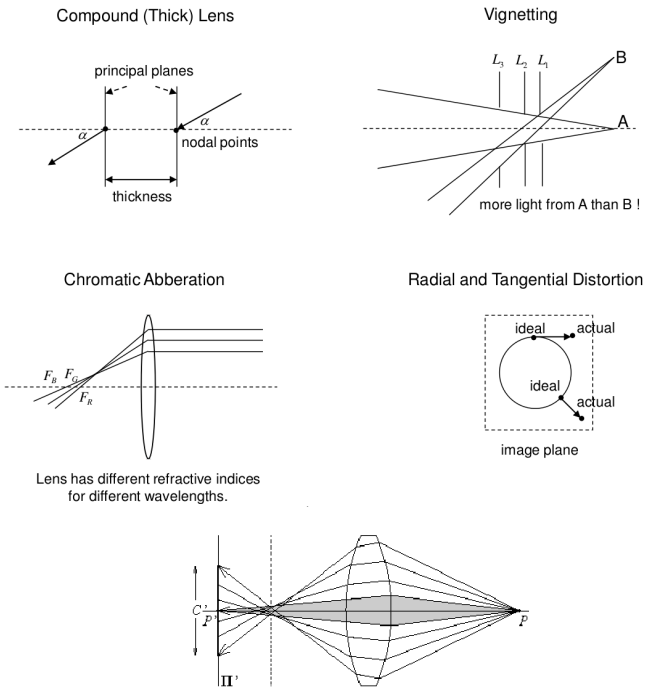


Figure 21: Some selected problems with lenses.

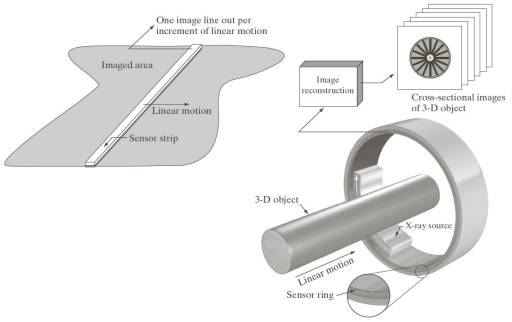


Figure 22: 1D and 3D sensing devices.

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.

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

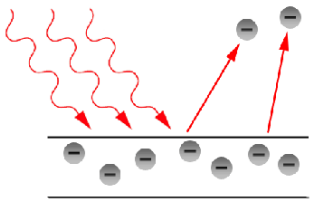


Figure 23: Photoelectric effect.

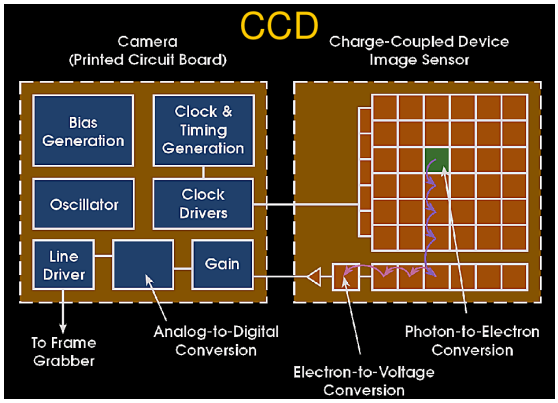


Figure 24: CCD sensor.

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).

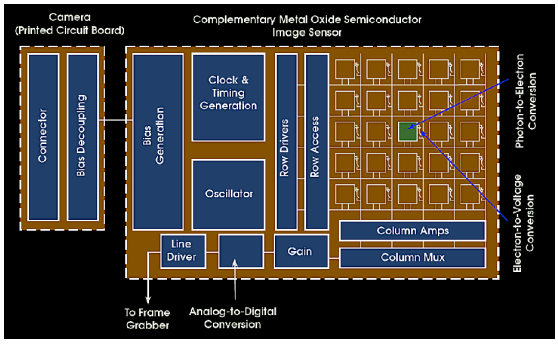


Figure 25: CMOS sensor.

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).

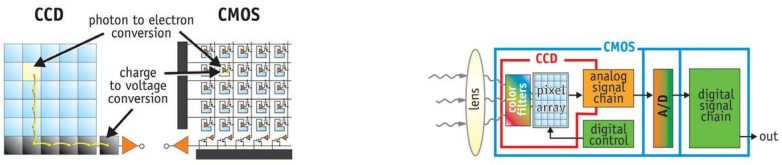


Figure 26: Schematic comparison of CCD and CMOS sensors.

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

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

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 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.

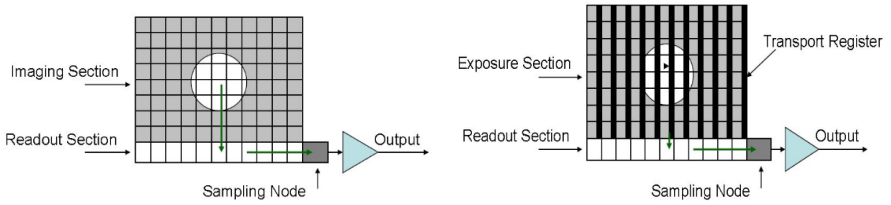


Figure 28: FF and IT CCD configurations.

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.

Figure 29: Smearing and blooming effects in CCDs configurations.

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 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.

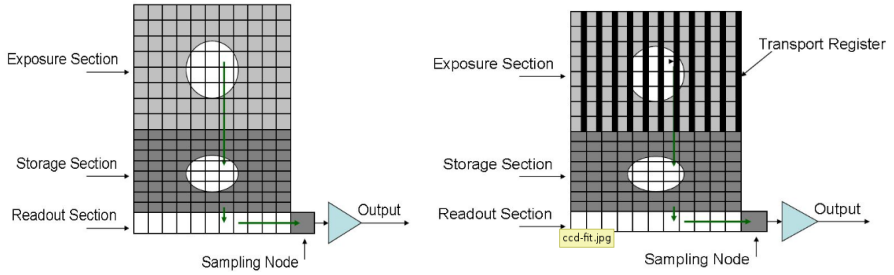


Figure 30: FT and FIT CCD configurations.

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 re-

duces 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.

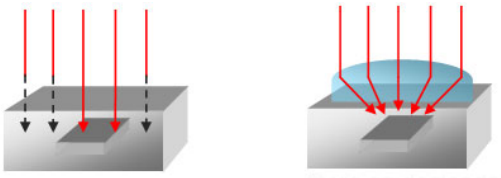


Figure 31: Microlens principle.

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):

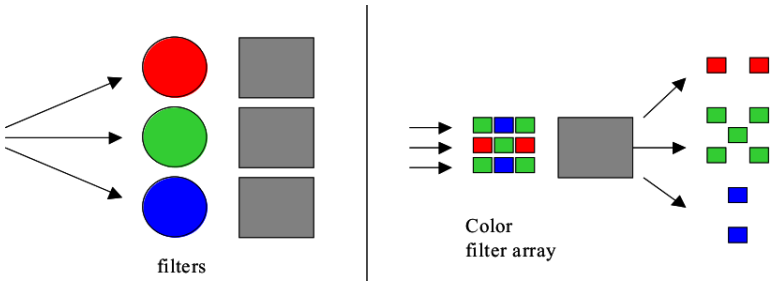


Figure 32: Approaches to colour sensing.

- 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.

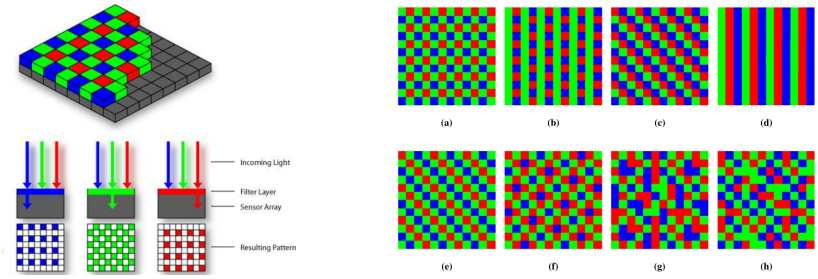


Figure 33: CFA examples.

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 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.

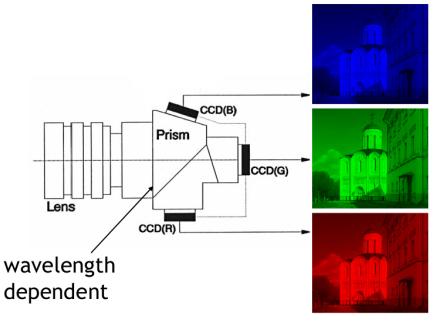


Figure 34: Non-spatial multiplexing colour sensing (multichip system.

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, 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.).

Fucussing done in cameras is fundamentally different as compared to the HVS – the lense does not change its shape (due to restrictive material properties)

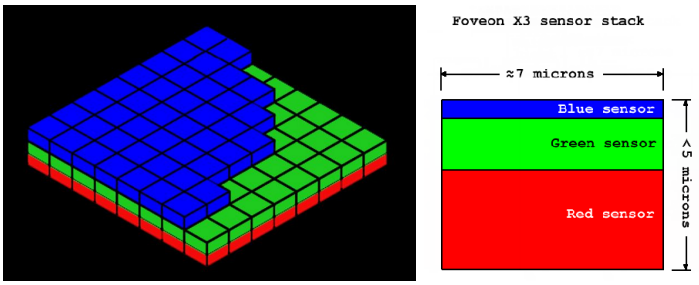


Figure 35: Non-spatial multiplexing colour sensing (Foveon system).

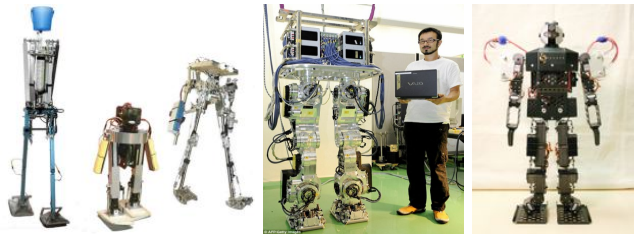


Figure 36: Means of transportation following the classical digital camera approach.

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.

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 37: source image

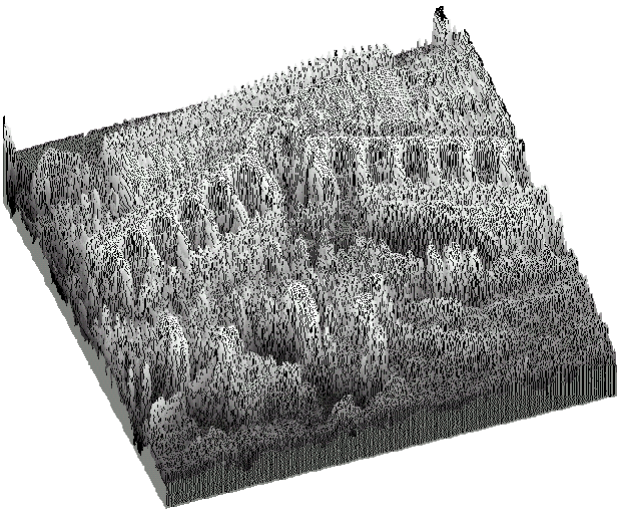


Figure 38: image as 2D-set of brightness values

Figure 39: source image

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.

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

red channel green channel blue channel

Figure 40: RGB decomposition of figure 39

luminance channel (Y) red-green balance (U=B-Y) yellow-blue balance (V=R-Y)

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 Rasterung und Quantisierung

- Unter Rasterung (siehe figure 42) versteht man die Aufteilung des Bildes (Diskretisierung) in festgelegte Abstände (Anzahl Pixel in beide Richtungen).
- Unter Quantisierung (siehe figure 43) versteht man die Bewertung der Hellligkeit eines Pixels mittels einer festgelegten Grauwertmenge (Bit pro Pixel bpp).
- Der Speicherbedarf für ein Bild errechnet sich aus $b = N \cdot M \cdot \ln(F)$.

- Die Auflösung des digitalen Bildes (Grad an unterscheidbarem Detail) hängt maßgeblich von den Parametern N, M und F ab.

Anmerkung: N, M ... Höhe und Breite (in Pixeln); F ... Anzahl der Grauwerte

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

Figure 42: Rasterung von figure 39

256 colors 64 colors 16 colors

Figure 43: Quantization 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.

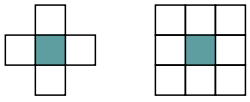


Figure 44: pixel neighborhoods

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).

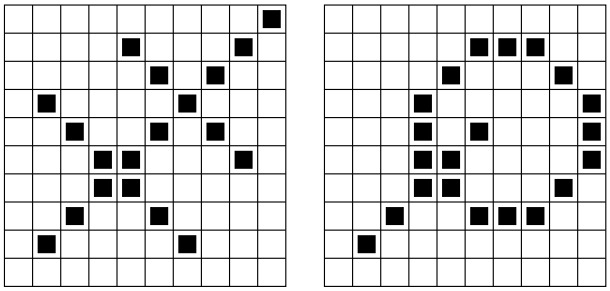


Figure 45: contiguity paradoxes

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

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.

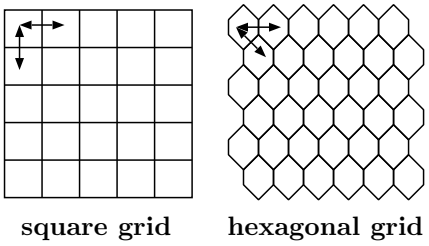


Figure 46: grid types

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.

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

2.3.4 Histograms

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.

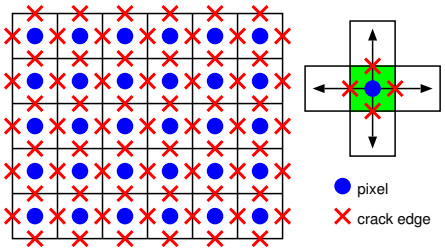


Figure 47: crack edges

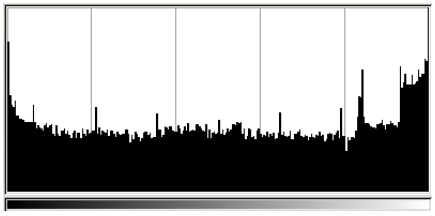


Figure 48: image histogram for figure 39

Figure 49: images with same histogram

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:

0000776655555566000006444444442221111112234444565221 1

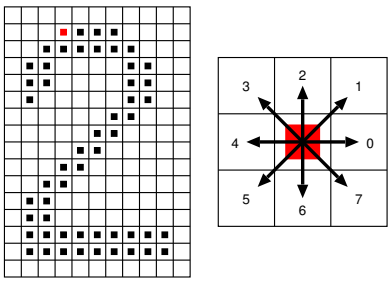


Figure 50: chain data structure

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.

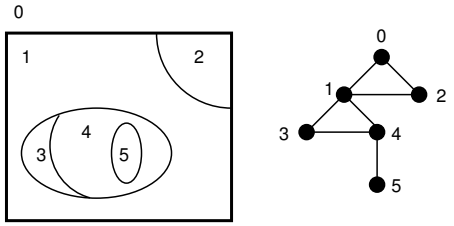


Figure 51: region adjacency graph

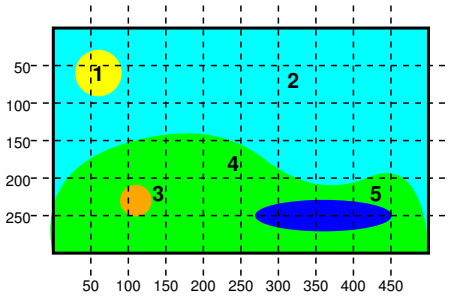
2.5.5 Relational Structures

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.

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.



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

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.

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

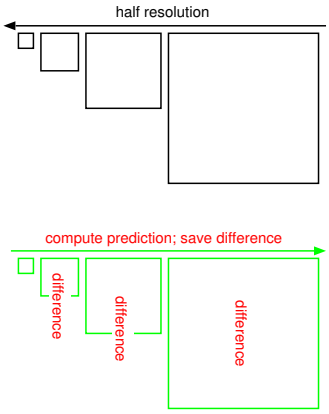


Figure 53: image pyramids

- 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).

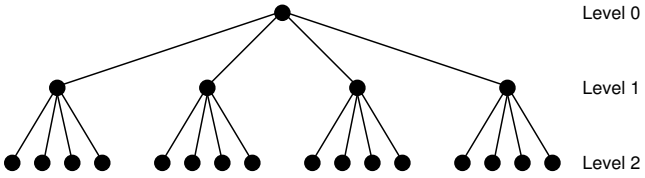


Figure 54: T-pyramid data structure

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.

2.7 Perception

Die menschliche Wahrnehmung entspricht dem Human Visual System (HVS).

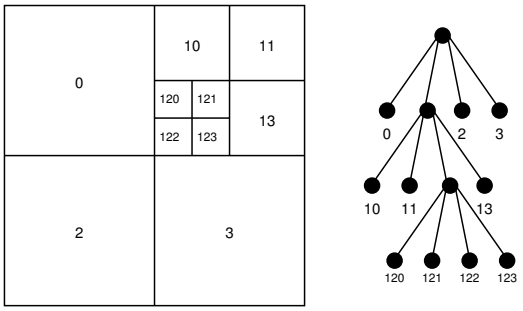


Figure 55: quadtree

Definition 1 (Kontrast) *entspricht der lokale Veränderung der Helligkeit, definiert als Quotient zwischen dem Durchschnitt der Objekthelligkeit und der Hintergrundhelligkeit.*

Der Kontrast ist eine logarithmische Eigenschaft (d.h.: keine Übereinstimmung von Numerik und Wahrnehmung; bei höherer Helligkeit braucht man für gleichen Eindruck höheren Kontrast).

Die **Sehschärfe** (acuity) ist am besten gegenüber mittleren Wechseln in der Helligkeit; weniger gut bei zu schnellen oder zu langsamen Wechseln und intersubjektiv verschieden → "Multiresolution Wahrnehmung"

3 Image Enhancement

Ziel des Image Enhancements ist es, Bilder vorzuverarbeiten, damit sie für spezielle Applikationen besser geeignet sind. Man unterscheidet dabei zwischen:

- spatial domain methods (Bildraum)
- transform domain (e.g., Frequenzraum-Fourier, Orts/Frequenzraum-Wavelets)

3.1 Spatial Domain Methoden

Es sei $f(x, y)$ die Bildfunktion des Ausgangsbilds und $g(x, y) = T(f(x, y))$ das *verbesserte* Bild. T steht für eine Operation auf $f(x, y)$ in einer Nachbarschaft[2] von (x, y).

Einfachster Fall: 1×1 Nachbarschaft, d.h. g hängt nur vom Wert von f an der Stelle (x, y) ab. T heisst dann **Grauwerttransformation** (siehe figure 56) mit der Form $s = T(v)$ mit $v = f(x, y)$ und $s = g(x, y)$

In der Graphik bezeichnet die x-Achse die ursprünglichen Werte, die y-Achse die Werte nach der Transformation.

[2] i.A. eine quadratische Bildregion mit Zentrum in (x, y). Das Zentrum wird von Pixel zu Pixel bewegt.

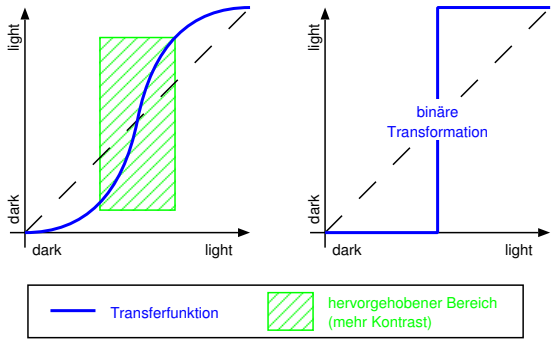


Figure 56: Grauwerttransformation

Größere Nachbarschaft: verschiedene Verarbeitungsfunktionen werden oft als **Masken** (Templates, Windows, Filter) bezeichnet. Ein kleines Array (e.g. 3×3 Pixel) wird über das Bild bewegt. Koeffizienten des Arrays sind so gewählt, dass gewisse Bildeigenschaften hervorgehoben werden.

Beispiel: Bild konstanter Intensität, mit isolierten Punkten anderer Intensität ("Pop Noise"); Maske: $w_i = -1\, i = 1, \ldots 9$ ausser $w_5 = 8$, jedes Feld wird mit darunterliegenden Pixeln multipliziert und die Ergebnisse addiert. Wenn alle Pixel gleich sind ist das Ergebnis 0. Die Maske wird Pixel für Pixel über das Bild geschoben (siehe figure 57).

$$\text{Ergebnis} \begin{cases} = 0 & \text{alle Pixel identisch} \\ > 0 & \text{Mittelpixel hat höheren Wert} \\ < 0 & \text{Mittelpixel hat geringeren Wert} \end{cases}$$

	...	$x-1$	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: Bild und Maske

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

3.2 Contrast Manipulierung & Modifizierung

3.2.1 Amplitudenveränderungen

Grauwertbereich verändern (siehe figure 58: Visualisierung von Differenzbildern nach Prediction oder MC; clipping wird gern verwendet wenn wenig Pixel in den Randbereichen des Histogramms liegen - Kontrast wird zusätzlich verbessert).

Anwendung: Alle hier besprochenen Verfahren kann man auch lokal auf Bildteile anwenden.

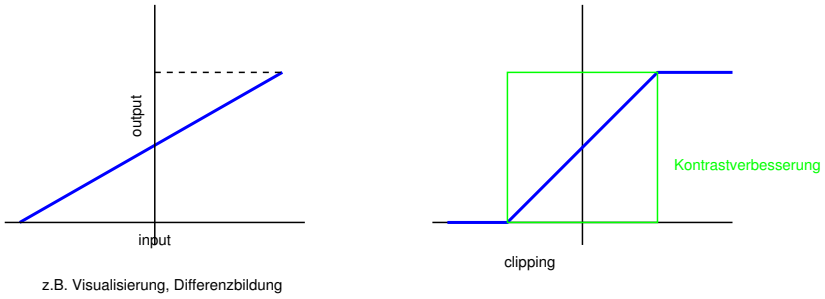


Figure 58: Grauwertbereichs Modifikation

3.2.2 Kontrast Modifikationen

Der Kontrast der Helligkeitsbereiche wird dort erhöht, wo die Steigung der Transferfunktion (oder ihrer Tangente) grösser als 1 ist.

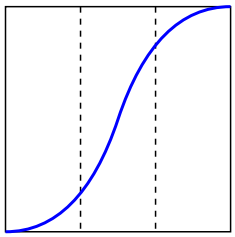


Figure 59: Kontrast Modifikation

Als Beispiel die Kontrast-Modifikation einer Computer Tomographie durch Anwendung des Logarithmus in 60.

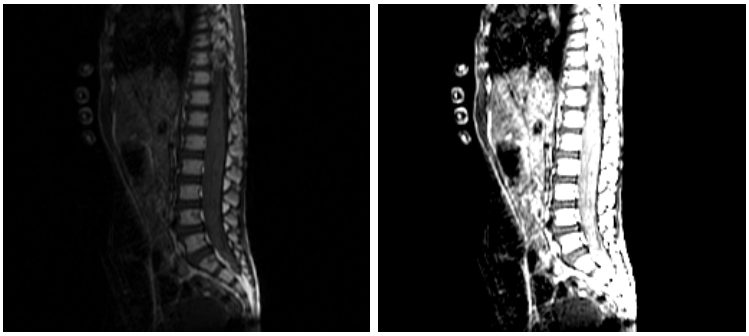


Figure 60: Kontrast Modifikation einer CT

Der Kontrast kann durch einfache Transferfunktionen wie

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

modifiziert werden (siehe auch Figure 59). Typische Kontrastmodifikationen sind in Figure 61 zu sehen.

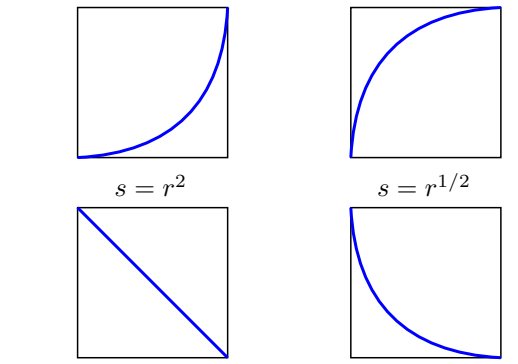


Figure 61: Typische Kontrastmodifikationen

Als Beispiel die Kontrast-Modifikation eines Myelins durch Anwendung des Logarithmus (ähnlich $s = r^{1/2}$) in 62.

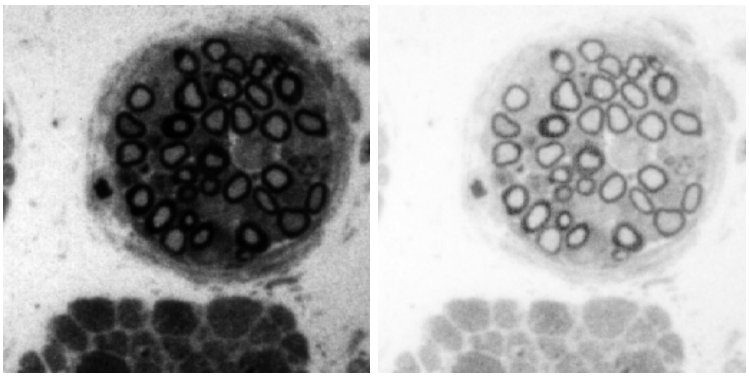


Figure 62: Kontrast Modifikation eines Myelins

3.2.3 Histogrammmodifizierung

Histogramm: Häufigkeitsverteilung der Grauwerte (globale Bildbeschreibung).

Sei r der Grauwert eines Pixel und $0 \leq r \leq 1$ mit $r = 0 =$ schwarz und $r = 1 =$ weiß. Wir betrachten die Transformation $s = T(r)$ mit den Eigenschaften

(a) T ist monoton wachsend auf $(0, 1)$

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

Umkehrtransformation von s nach r ist $r = T^{-1}(s)$ mit $0 \leq s \leq 1$ mit den analogen Bedingungen (a) und (b).

Betrachte die Grauwerte als stetige Zufallsvariable. Man kann die originale und transformierte Grauwertverteilung durch ihre Wahrscheinlichkeitsdichten $p_r(r)$

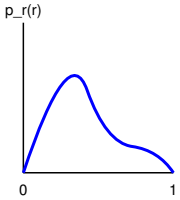


Figure 63: stetiges Histogramm

und $p_s(s)$ darstellen. Die Dichten geben Aussagen über das Grundaussehen des Bildes.

Anmerkung: Diese Wahrscheinlichkeitsdichten bilden ein *stetiges Histogramm*.

Aus der elementaren Wahrscheinlichkeitstheorie:
Kennt man $p_r(r)$, $T(r)$ und $T^{-1}(s)$ genügt Bedingung (a), so gilt für die Dichte der transformierten Grauwerte:

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

3.2.4 Histogramm-Equalisierung

Betrachte die Transferfunktion (kumulative Verteilungsfunktion):

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

Durch Ableiten nach r erhält man (Hauptsatz der Differential und Integralrechnung):

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

Setzt man nun die Gleichung (2) in Gleichung (1) ein so erhält man:

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

Das Ergebnis ist eine gleichmässige Dichtefunktion, konstant 1. Interessanterweise ist dieses Ergebnis unaghängig von der inversen Funktion.

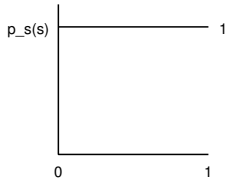


Figure 64: equalisiertes Histogramm

Eine Histogramm-Equalisierung wird also durch Anwendung der kumulativen Verteilungsfunktion als Transferfunktion durchgeführt. Bei einem equalisierten Histogramm sind dann alle Auftritts-Wahrscheinlichkeiten 1 (Achtung: hier sind wir im – *idealen* – stetigen Fall !.

Beispiel:

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

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

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

Problem: Bei echten Bildern gibt es keine "Funktion" $p_r(r)$.

Diskretisierung

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

n_k ... Anzahl wie oft k-ter Grauwert auftritt
n ... Anzahl der Pixel
L ... Anzahl der Grauwerte

Diskrete Equalisierung

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

Bemerkungen: Man braucht keine Inverse. $T(r_k)$ kann aus Statistik gewonnen werden. Durch die Diskretisierung ist das Ergebnis nur eine Approximation!

Ein durch Histogramm Equalisierung verbessertes Bild kann in figure 65 betrachtet werden. Zu dem Ausgangsbild und dem Histogramm-equalisierten Bild sind auch noch die Histogramme gegeben.

3.2.5 Direkte Histogrammspezifikation

Seien $p_r(r)$ und $p_z(z)$ die originale und gewünschte Dichtefunktion. Das Bild wird im ersten Schritt Histrogramm-Equalisiert

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

Wäre das gewünschte Zielbild verfügbar, könnte man es auch equalisieren

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

$z = G^{-1}(v)$ gäbe wieder die gewünschten Pixelwerte.

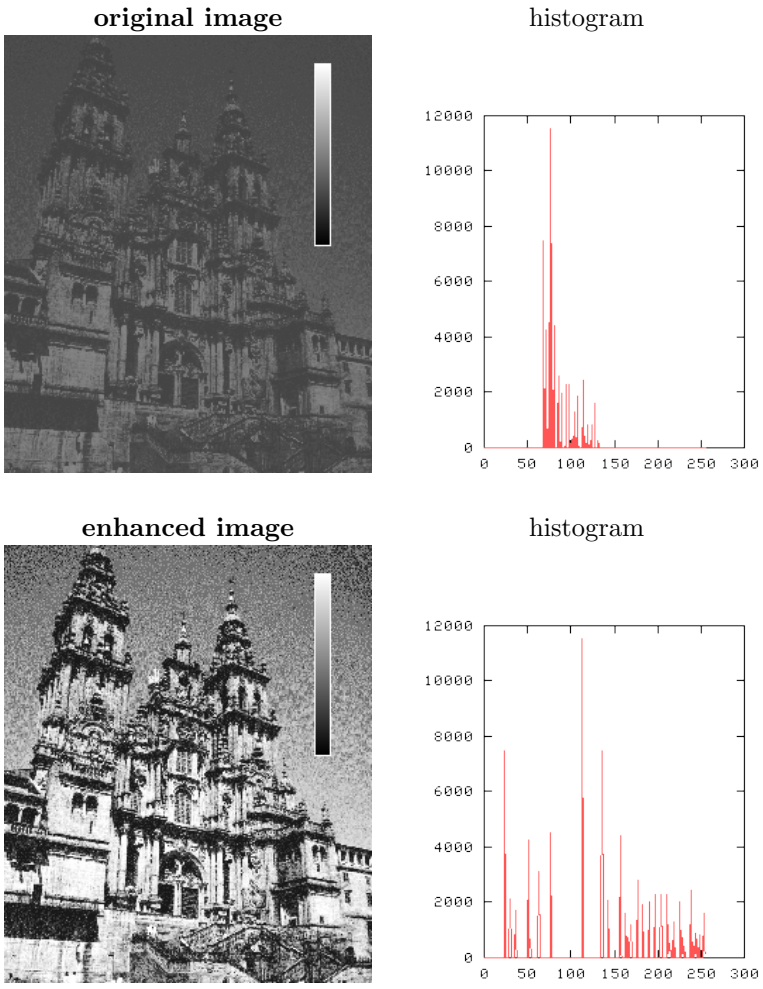


Figure 65: Histogramm Equalisierung

$p_s(s)$ und $p_v(v)$ haben identische uniform density[3]. Man benützt daher anstelle von v im inversen Prozess die Werte s (vom equalisierten Original). Somit hat $z = G^{-1}(s)$ die gewünschte Dichte.

Vorgehensweise:

1. Equalisiere Originalbild $\rightarrow s$
2. Spezifiziere die gewünschte Dichte und erhalte $G(z)$
3. $z = G^{-1}(s) \Rightarrow z = G^{-1}(T(r))$

Problem: Die Umkehrfunktion kann (im Diskreten) nicht analytisch berechnet werden; wird durch ein Mapping von Grauwert auf Grauwert erhalten.

Anwendung: Optimierung für bestimmte Devices, deren Charakterisierung man kennt.

Bemerkung: Die besprochenen Verfahren können auch lokal auf $n \times m$ Nachbarschaften angewendet werden – wenn nur eine bestimmte Region interessant ist liefert das dort bessere Ergebnisse.

3.3 Bildglättung (image smoothing) & Denoising

Ziel: Effekte von Transmission oder Samplingfehler sollen korrigiert werden. Diese Effekte sind **lokale Störungen** (im Idealfall einzelne Pixel)!

Die am meisten verwendete Methode ist das im folgenden beschriebene *Neighbourhood Averaging*.

3.3.1 Neighbourhood Averaging

$g(x, y)$ wird erhalten über die Durchschnittbildung in einer Umgebung S:

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

M ... Anzahl der Pixel in S

Auch hier können ähnlich figure 57 Masken für die Umgebung verwendet werden:

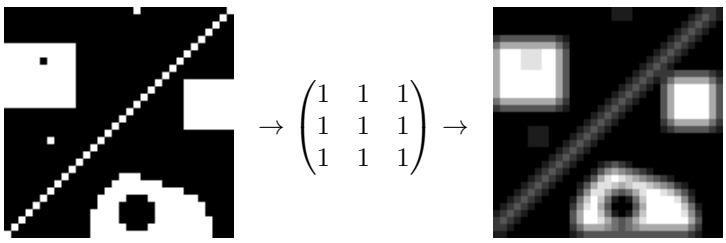


Figure 66: Averaging mit 3×3 Maske

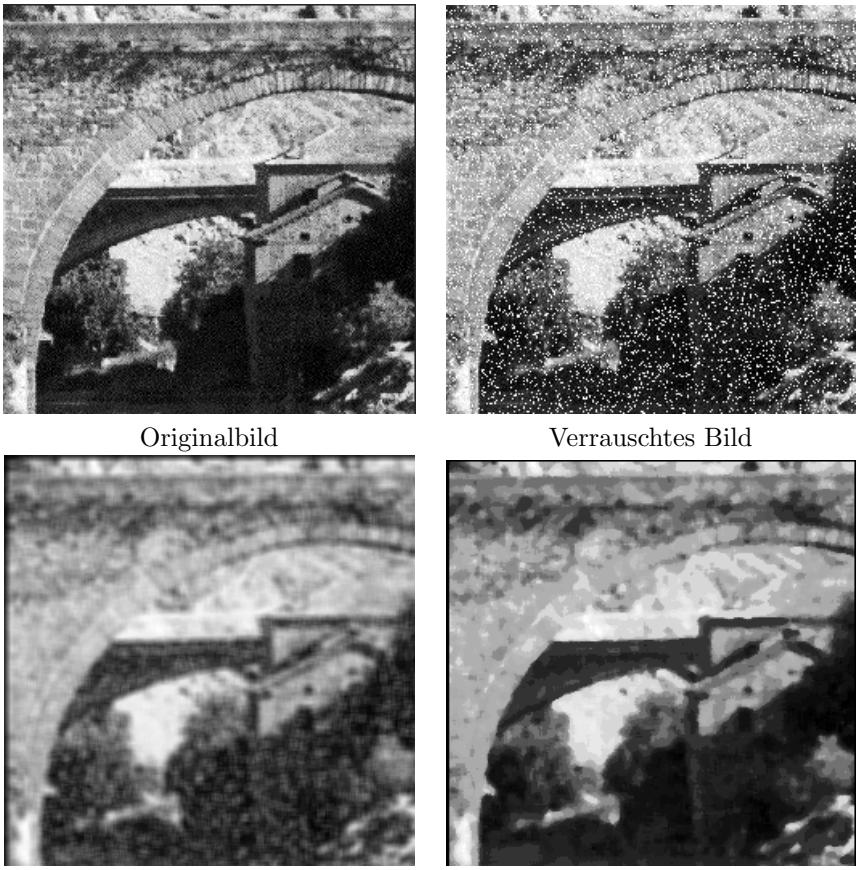
[3] gleichmäßige Dichte

Nachteil: Kanten werden extrem verwischt (*blurring*)! Allerdings kann man durch thresholding (mit threshold T) Abhilfe schaffen. Bei zu grossen Unterschieden zwischen Original- und Durchschnittswert unterbleibt die Durchschnitts-Bildung und der Originalwert bleibt erhalten.

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

3.3.2 Median Filtering

Anstelle der Mittelwert-Bildung in der Nachbarschaft wird der Median verwendet. Dadurch beeinflussen statistische *outlyer* das Ergebnisse nicht. Für Denoising (insbes. geg. pop-noise) besser geeignet als Averaging (siehe Fig. 67).



Originalbild | Verrauschtes Bild

5 x 5 Averaging | 5 x 5 Median

Figure 67: Denoising

3.4 Image Sharpening

Beim *image sharpening* sollen Kanten hervorgehoben werden;
Idee: Differenz zwischen Pixeln deutet auf Existenz einer Kante.

Averaging und sharpening liegen zwei unterschiedliche Gedanken zu Grunde. Während beim averaging versucht wird Details zu "integrieren" werden beim sharpening Details "differenziert".

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

1. G zeigt in die Richtung des größten Anstiegs von $f(x,y)$
2. $|G[f(x,y)]| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \sim \text{mag}(G)$

$\text{mag}(G)$ ... Größe (magnitude) von f, gibt maximale Wachstumsrate von $f(x,y)$ an.

In der Bildverarbeitung wird $\text{mag}(G)$ selbst oft als Gradient bezeichnet.

Diskretisierung
Ableitungen werden durch Differenzen approximiert:

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

Alternativ können auch Absolutbeträge verwendet werden (effizientere Implementierung).

Roberts Operator (siehe auch Kapitel 5.1.1) und Fig. 68).

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

I.a. ist der Gradientenwert proportional zur Differenz im GW zwischen benachbarten Pixeln – grosse Werte für Kanten, kleine für glatte Bereiche.

Es gibt verschiedene Möglichkeiten das *Gradientenbild* $g(x,y) = |G[f(x,y)]|$ zu visualisieren:

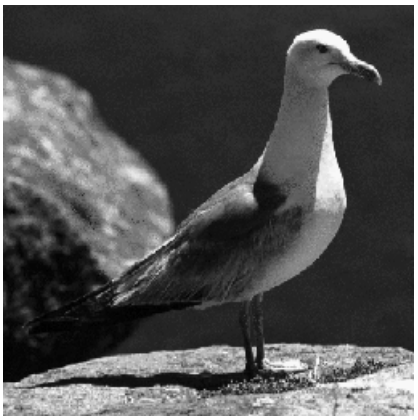
3.5 Methoden im Transformationsbereich

Die verwendeten Transformationen sind unitäre[4] Transformationen und werden eingesetzt für:

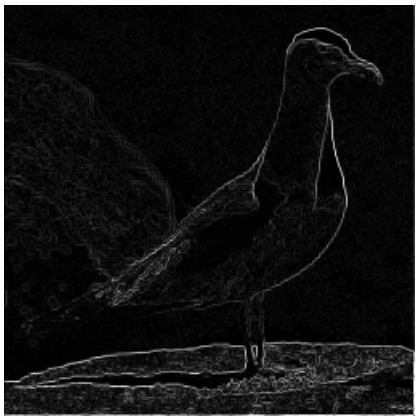
Feature extraction um bestimmte Merkmale effizient beschreiben zu könenn (z.B. Frequenzen: hoch - Kanten, nieder - Helligkeit). Ziel ist es bestimmte Operationen anhand solcher Merkmale besser durchführen zu können (z.B. Denoising).

Kompression zur Konzentration von Information

[4] orthogonal und regulär



Originalbild



Roberts Gradientenbild

Figure 68: Image Sharpening

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

Figure 69: Arten des Gradientenbilds

Efficient calculations z.B. wird eine *dense matrix* in eine *sparse matrix* überführt, da es für sparse matrices effizientere Algorithmen gibt (in sparse matrices – dünn besetzten Martizen – sind viele Koeffizienten null).

Meistens wird das Konzept verwendet ein Signal mit Hilfe von orthogonalen Basisfunktionen darzustellen.

Hintergrund: Vektoren im 2 dimensionalen Raum können durch einen Satz von orthogonalen (Def.: inneres Produkt ist 0) Basis-Vektoren dargestellt werden (Orthogonalbasis):

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

$\{(1, 0), (0, 1)\}$ sind die orthogonalen Basisvektoren. α und β sind die Koeffizienten die angeben, wie stark jeder Basisvektor verwendet werden muß um den Vektor (x, y) darstellen zu können. Nur die Orthogonalitätseigenschaft ermöglicht eine minimale Anzahl von Basisvektoren.

Dieses Konzept kann auf Funktionen bzw. Signale übertragen werden.

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

Die $\psi_n(x)$ sind die orthogonalen Basisfunktionen, $< f(x), \psi_n(x) >$ sind die Transformationskoeffizienten die angeben, wie stark jede Basisfunktion verwendet werden muß um das Signal "gut" darstellen zu k"onnen. F"ur eine Anwendung werden die $< f(x), \psi_n(x) >$ berechnet und dann weiterverarbeitet. Da die $\psi_n(x)$ orthogonal sind, ist die entsprechende Anzahl minimal.

z.B.: im Falle der Fouriertransformation (s.u.) sind die $\psi_n(x) = e^{-\pi inx} = \cos(nx) - i\sin(nx)$. In diesem Fall werden Frequenzen von periodischen Signalen betrachtet. Ein Fourierkoeffizient $< f(x), \psi_n(x) >$ gibt die Stärke des Frequenzanteils n in einem Signal an. Es ist klar, daß nicht alle Signale auf diese Art am effizientesten dargestellt werden können.

3.5.1 Fouriertransformation

Entwickelt vom Franzosen Fourier, der sich viel mit Musik (Geige) beschäftigt hat und wissen wollte, wie Töne durch Verkürzung von Seiten entstehen. Für mehr Hintergrund siehe z.B. `http://de.wikipedia.org/wiki/Fourier-Transformation` .

Sei $f(x)$ eine stetige Funktion, dann ist $\hat{f}(u)$ die Fouriertransformation von $f(x)$.

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

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

Die Umkehrung funktioniert, wenn $f(x)$ stetig und integrierbar ist und $\hat{f}(u)$ integrierbar ist.

Die Fouriertransformation einer reellen Funktion ist i.A. komplexwertig.

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

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

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

$|\hat{f}(u)|^2$ ... Power-Spektrum (Spektraldichte)
$|\hat{f}(u)|$ ... Fourier-Spektrum (Frequenzspektrum)
$\Phi(u)$ ... Phasenwinkel
u ... Frequenzvariable (da $e^{2\pi iux} = \cos 2\pi ux + i\sin 2\pi ux$)

Interpretiert man das Integral als Summation diskreter Terme, wird klar dass $\hat{f}(u)$ aus einer unendlichen Summe von Sinus- und Cosinus Termen zusammengesetzt ist, wobei der Parameter u die Frequenz des Sin/Cos Paares bestimmt.

Diskrete Fourier Transformation (DFT) $\{f(0), f(1), \ldots, f(N-1)\}$ seien N gleichmäßig abgetastete Werte einer stetigen Funktion. Die DFT lautet dann

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

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

Zweidimensional

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

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

Für Display-Zwecke ist $D(u,v) = \log(1 + |\hat{f}(u,v)|)$ besser geeigent als $|\hat{f}(u,v)|$, da die Werte bei steigender Frequenz stark abnehmen.

Ein paar Beispiele von DFT Transformationen sind zu sehen in figures 70 und 71.

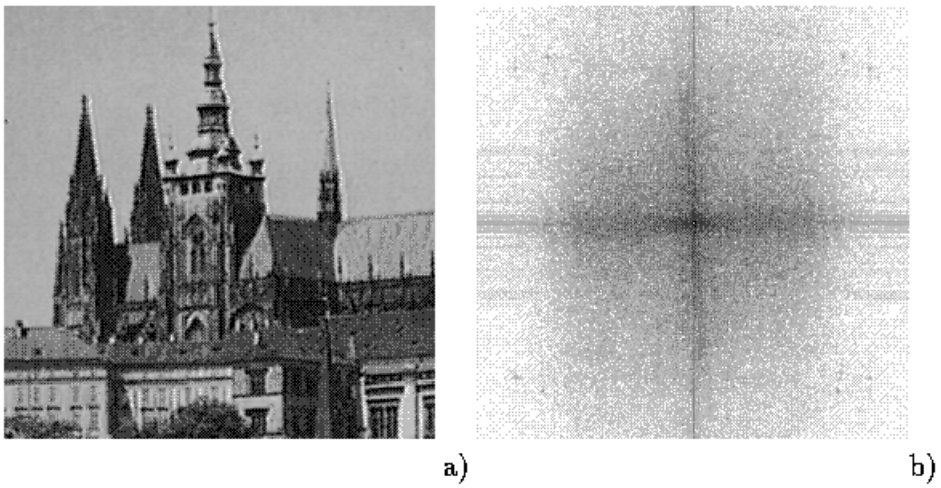


Figure 70: Origial image and its Fourier Spectrum (Magnitude)

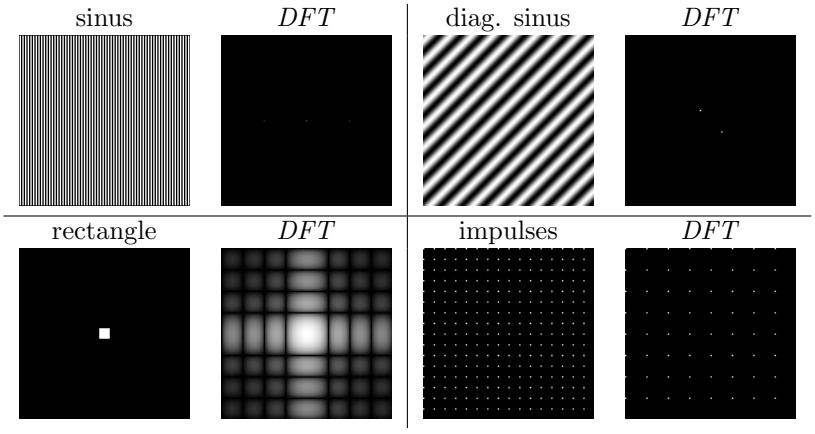


Figure 71: DFT Transformations

Eigenschaften der 2D Fourier Transformation

- $\hat{f}(0,0)$ entspricht dem durschnittlicher Grauwert über alle Pixel

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

- **Separabilität**

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

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

  Die zweidimensionale Transformation ist realisierbar als Hintereinanderreihung von zwei eindimensionalen Fourier Transformationen, d.h. zuerst DFT auf alle Zeilen, dass DFT auf alle Spalten. Grundlage ist die Separierbarkeit in den Basisfunktionen, d.h. $e^{-2\pi i(ux+vy)} = e^{-2\pi i u x} e^{-2\pi i v y}$.

- **Translation**

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

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

  Sei $u_0 = M/2$ und $v_0 = N/2$

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

  Somit kann der Ursprung der Fourier Transformation $(0,0)$ zum Zentrum der Frequenzebene $(M/2, N/2)$ bewegt werden durch Multiplikation von $f(x,y)$ mit $(-1)^{x+y}$ **und** ein Shift in $f(x,y)$ läßt $|\hat{f}(u,v)|$ unverändert (Shiftinvarianz der DFT).

$$|\hat{f}(u,v) e^{-2\pi i(u x_0/M + v y_0/N)}| = |\hat{f}(u,v)|$$

- **Periodizität**

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

- **Symmetrie** Falls $f(x,y)$ reell:

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

  Durch die konjugierte Symmetrie um den Ursprung ist die Hälfte der Transformationsmethoden redundant. Symmetrie um den Ursprung und Periodizität ermöglicht es um die volle Periode zu erhalten den Ursprung des Transformationsbereichs mittels Translation nach $(M/2, N/2)$ zu legen (wie: siehe vorher).

- **Linear Kombination**

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

- **Skalierung**

  $af(x,y) = a\hat{f}(u,v)$ im Gegensatz zu $\quad f(ax,by) = \frac{1}{ab}\hat{f}(u/a, v/b)$

  Skalierung ist wie folgt einzusehen (1-dim.): $\hat{f}(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iux}dx$ für $f(x)$. Für f(ax) gilt analog $\hat{f}(u) = \int_{-\infty}^{\infty} f(ax)e^{-2\pi iux}dx$. Multiplikation des Integrals und des Exponenten mit a/a ergibt $1/a \int_{-\infty}^{\infty} f(ax)e^{-2\pi iax(u/a)}adx$. Durch die Variablensubstitution $s = ax$ $(ds = adx)$ erhalten wir $1/a \int_{-\infty}^{\infty} f(s)e^{-2\pi is(u/a)}ds$. Dieser Ausdruck ist offensichtlich $\frac{1}{a}\hat{f}(\frac{u}{a})$. Eine kontrahierte Funktion $(a > 1)$ hat also demnach eine Fouriertransformierte mit reduzierter Amplitude und horizontaler Streckung im Frequenzbereich.

Laplacian

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

Faltung Die Faltung der Maske $h(x)$ auf das Bild $f(x)$ ist wie folgt definiert

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

Faltungssatz

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

$f(x) * g(x)$ ... steigende Komplexität mit Größe der Maske f.
$\hat{f}(u) \cdot \hat{g}(u)$ ... keine steigende Komplexität wenn Maske f bekannt

Der Faltungssatz findet seine Anwendung in ...

Komplexitätsreduktion bei Faltung Fourier Transformation von f und g berechnen und multiplizieren (rentiert sich erst bei Masken ab 20^2)

Filterungen im Frequenzbereich (siehe Kapitel 3.5.2)

Fast Fourier Transformation (FFT) Die FFT wurde 1968 von Cooley und Tuckey entwickelt und beruht auf einer Idee von C.F. Gauss. Sie wurde entwickelt, weil bei N zu transformierenden Punkten die Komplexität der DFT $\mathcal{O}(N^2)$ zu hoch war. Die FFT verringert dies auf $\mathcal{O}(N \log N)$ und macht erst die Verwendung in der Signalverarbeitung möglich.

3.5.2 Filterung im Frequenzbereich

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

$\hat{h}(u,v)$ ... Transferfunktion
$g(x,y)$ ... Shiften einer Maske $h(x,y)$ über das Bild $f(x,y)$

Vorgehensweise ($f(x,y)$ ist gegeben)

- Berechnung von $\hat{f}(u,v)$
- wähle $\hat{h}(u,v)$ so, daß das entstehende Bild bestimmte Eigenschaften hervorhebt
- Bild durch inverse Fourier Transformation von $\hat{h}(u,v) \cdot \hat{f}(u,v)$ berechnen

In figure 72 sind die im folgenden beschriebenen Filter dargestellt.

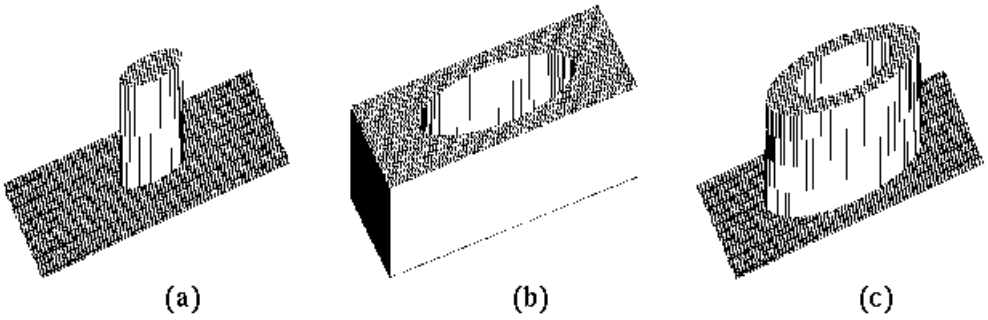


Figure 72: Verschiedene Filter

Lowpass Filter Kanten und scharfe Übergänge sind hochfrequente Phänomene. Schwächt man diese Teile im Frequenzbereich so erreicht man eine Bildglättung.
$\hat{h}(u,v)$ sei der *Ideal Lowpass Filter* (ILPF)

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

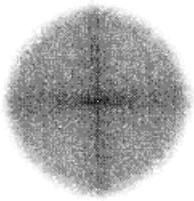
D_0 ist die sog. Cut-off Frequenz und $D(u,v) = (u^2 + v^2)^{1/2}$ die Entfernung von (u,v) vom Ursprung. Durch Anwendung von $\hat{h}(u,v) \cdot \hat{f}(u,v)$ werden alle höherfrequenten Bereiche (Kanten) 0, die niederfrequenten Bereiche bleibene erhalten (siehe Fig. 73). Derartige Filter betreffen Real- und Imaginärteil und ändern die Phase nicht (*zero-phase shift*).

Probleme

- nicht in Hardware realisierbar
- durch scharfe 0-Setzung entstehen Artefakte (*ringing*)

  Das Aussehen von $h(x,y)$ hängt von D_0 ab. Die Radien der Ringe sind invers proportional zum Wert von D_0 (d.h.: kleines D_0 erzeugt kleine Anzahl von breiten Ringen *starkes ringing*). Wächst D_0 so steigt die Anzahl der Ringe und ihre Breite nimmt ab.

a) b)



c) d)

Figure 73: ILPF

Butterworth Filter (BLPF)

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

Der Buttermorth Lowpass Filter ist eine Transferfunktion der Ordnung n. Sie hat keine scharfe Unstetigkeit und dadurch wenige Störungen.

Highpass Filter Analog zu den Lowpass Filtern werden bei den Highpass Filtern die hohen Frequenzen durchgelassen und somit Kanten und scharfe Übergänge betont.

$\hat{h}(u,v)$ sei der *Ideal Highpass Filter* (IHPF) (siehe Fig. 74).

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

Butterworth Filter (BHPF)

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

Der Butterworth Highpass Filter ist eine Transferfunktion der Ordnung n. Er beläßt nur Kanten im Bild und filtert alles andere heraus. Um auch Teile der niederen Frequenzen im Bild zu behalten kann man eine Konstante zur Transferfunktion addieren (*High Frequency Emphasis*). Zusätzlich verbessert z.B. Histogrammequalisierung das Ergebnis.

Bandpass Filter Hierbei wird ein bestimmte mittlerer Frequenzbereich *durchgelassen* und $\hat{h}(u,v)$ entsprechend definiert (Ergebnis siehe Fig. 75).

Speziellere Filterungsmethoden nehmen beispielsweise auf Charakteristika des der Bildstörung zugrundeliegenden Rauschens Rücksicht (siehe z.B. Fig. 76).

3.5.3 Wavelet Transformation

Motivation
Die Fouriertransformation kann per se nicht lokal Frequenzen filtern. Dafür verwendet man die **gefensterte Fouriertransformation**, welche das Signal in Stücke zerteilt und die Fouriertransformation auf die einzelnen Stücke anwendet. Aufgrund der fixen Fensterbreite gehen jedoch gewisse Frequenzen verloren. Eine umfassende Lösung stellt die Wavelet Transformation dar.

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

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

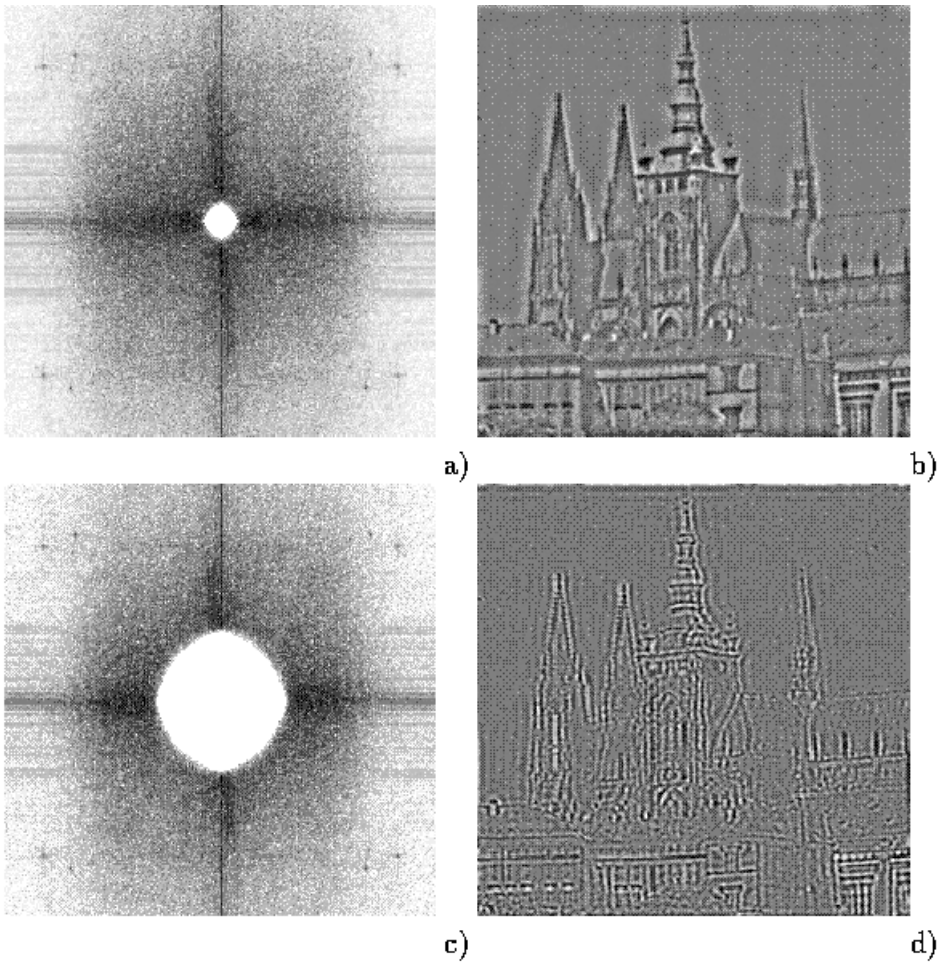
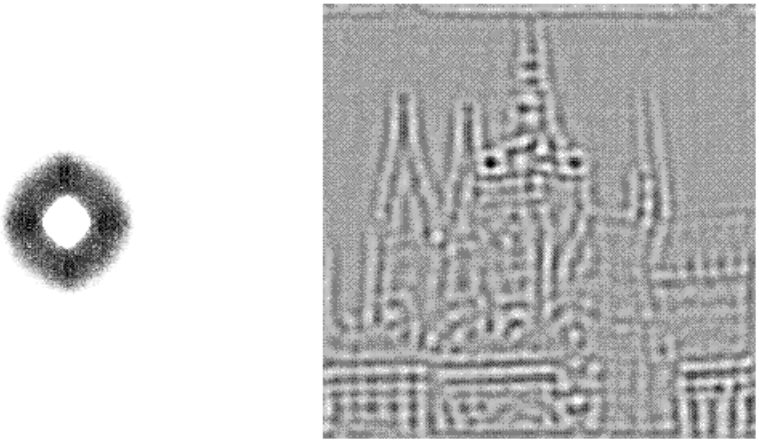
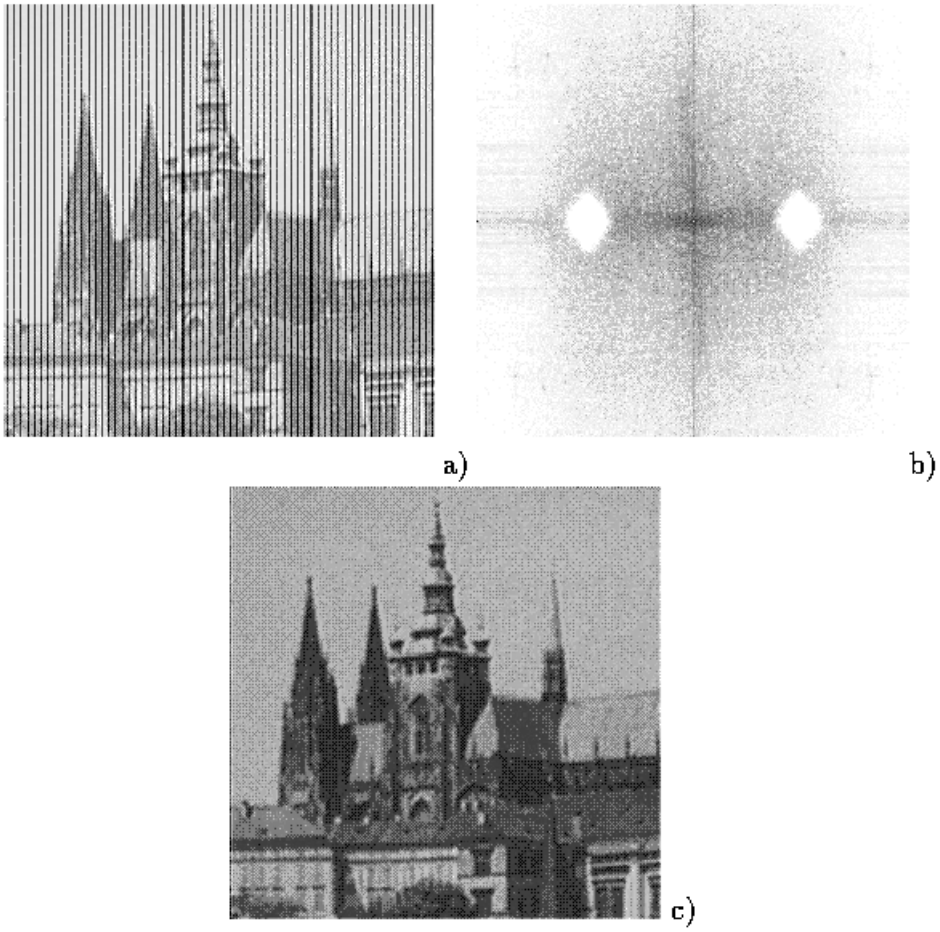


Figure 74: IHPF



a) b)

Figure 75: BPF



a) b)

c)

Figure 76: Filterung spezieller Frequenzbereiche

Die Funktionen aus Gleichung (17) werden Wavelets genannt, $\psi(s)$ wird oft *mother wavelet* genannt. Beispiele einiger mother wavelets sind:

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

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

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

Die Wavelet Transformation hängt von zwei Parametern (a und b) ab. Wenn sich a ändert, beschreiben die Wavelets aus Gleichung (17) unterschiedliche "Frequenzbereiche". Grössere a beschreiben breite, eher niederfrequente Funktionen, kleine eher schmale, hochfrequente. Wird der Parameter b geändert verschiebt sich das Zeit-Lokalisierungszentrum (welches in $s = b$ liegt). Alle Wavelets sind also verschobene und skalierte Versionen des mother wavelets.

Multiresolution Analysis Idee
Darstellung von MRA Signalen durch verschiedene Stufen der Approximation und den Unterschieden zwischen diesen Approximationen. Verwendet werden orthogonale Basisfunktionen, die Parameter a und b werden diskretisiert: $a = a_0^m$, $b = nb_0a_0^m$ mit $m, n \in \mathbf{Z}$ und $a_0 > 1$, $b_0 > 1$. Gern nimmt man $a_0 = 2$ and $b_0 = 1$.

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

Eine MRA wird durch ineinander verschachtelte Approximations- und Detailräume aufgespannt, die Funktionen $\phi(t)$ und $\psi(t)$ sind die jeweiligen (Orthonormal)basen.

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

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

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

Die "Scaling Equation" stellt eine Beziehung zwischen um Faktor zwei dilatierter Scaling Function und ihren ganzahling dilatierten Versionen dar (Funktionen niedererer Frequenz werden durch solche höherer dargestellt). Springender Punkt: die Folge $h(n)$ bestimmen die resultierenden Funktionen in eindeutiger Weise.

Fast Wavelet Transformation Eine Fast Wavelet Transformation im eindimensionalen Fall:

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

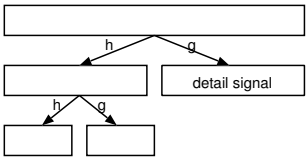


Figure 77: Wavelet Transformation

Möglichkeiten zur Randbehandlung bei ein-dimensionaler Berechnung:

- Periodisierung
- Fortsetzung
- Spiegelung
- Zero-Padding

2D Wavelet Transformation Bei Bildern ist die Transformation einer zweidimensionalen Funktion nötig. Dabei wird das Bild zuerst eindimensional zeilenweise und dann spaltenweise Wavelet transformiert. Dieses Verfahren wird meist noch mit downsampling (mit dem Faktor 2) kombiniert. Eine Veranschaulichung des Verfahres der zweidimensionalen Wavelet Transformation findet sich in figure 78. Ein Beispiel eines Wavelet transformierten Bildes ist in figure 81 zu sehen.

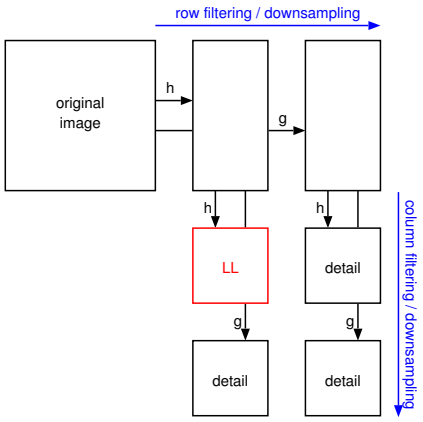


Figure 78: 2D Wavelet Transformation

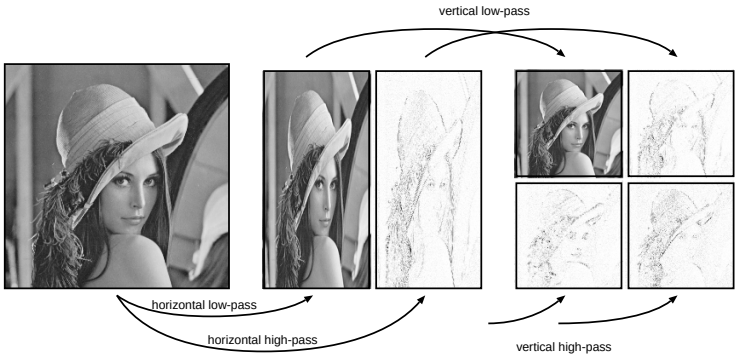


Figure 79: 2D Wavelet Transformation Visualisierung 1. Ebene

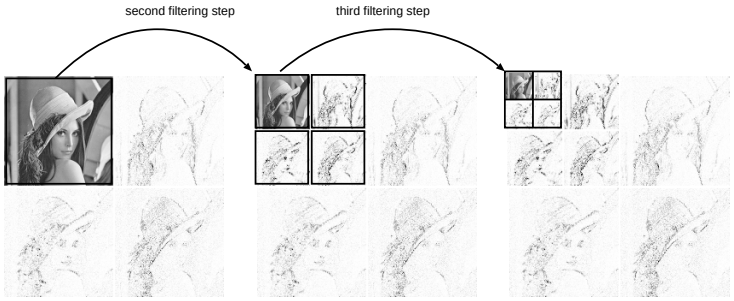


Figure 80: 2D Wavelet Transformation Visualisierung 2.+3. Ebene

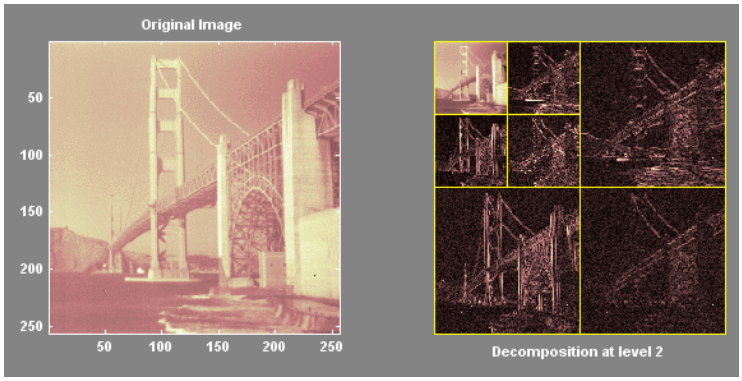


Figure 81: Wavelet Transformation Beispiel

Filterung im Waveletbereich

Lowpass Filter Detail-Subbands 0 setzen

Highpass Filter LL-Subband (und tiefe Detail-Subbands) 0 setzen

Bandpass Filter Subband, welches interessant ist als einziges nicht 0 setzen

Bemerkung: Wenn man das LL-Subband 0 setzt, so verschwinden die grundlegenden Bildinformationen.

Denoising Denoising wird durch Thresholding erreicht. Es bleiben nur Detailkoeffizienten über einer bestimmten Schranke (Threshold) erhalten.

Anwendung Die Wavelet Transformation wird zur Kompression in den folgenden Formaten verwendet:

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

Weiters werden Wavelets in der Signalanalyse und Bildanalyse verwendet.

3.5.4 Fourier vs. Wavelet

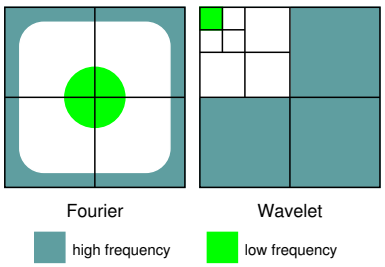


Figure 82: Fourier and Wavelet Transformation

Bei der Fourier-Transformation entspricht ein Koefizient dem globalen Frequenzgehalt des gesamten Bildes mit den Frequenzen u und v. Bei der Wavelet Transformation entspricht ein Koeffizient dem lokalen Frequenzgehalt 2^i an der entsprechenden Stelle im Bild. Aus diesem Grund werden bei Störungen im Frequenzband i.A. Fouriermethoden angewandt, wogegen man bei lokalen Störungen Waveletmethoden einsetzt. In figure 82 sind die Anordnung der Frequenzen nach einer der jeweiligen Transformation dargestellt.

3.5.5 Weitere Varianten der Wavelet Transformation

Wavelet Packet Transformation (WP) Grundlegendes: die iteration der Zerlegung wird nicht nur auf das low-pass subband angewendet sondern auf alle

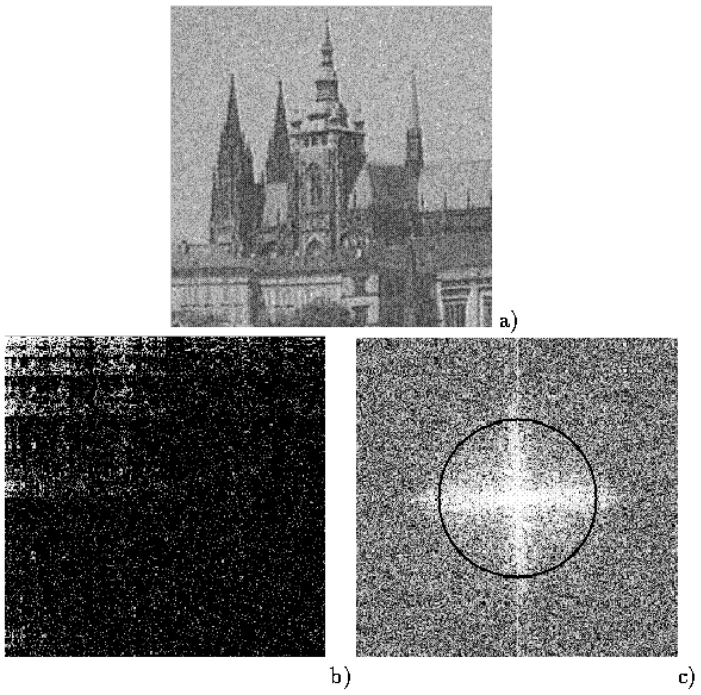


Figure 83: Denoising im Wavelet/Fourier Bereich

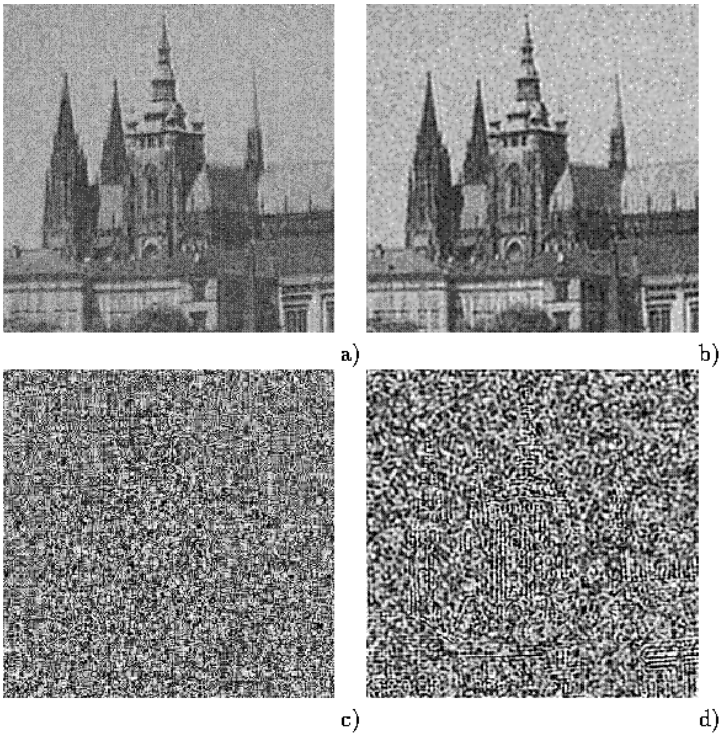


Figure 84: Denoising im Wavelet/Fourier Bereich: Ergebnisse

Freuquenzbänder. Dadurch wird eine bessere Frequenzauflösung erreicht (v.a. im höherfrequenten Bereich).

Best Basis: ist eine Methode den Teilbaum zur Repräsentierung zu verwenden, der das Bild am kompaktesten darstellt. Anwendung Kompression (z.B. FBI-standard, J2K Part II), Bäume durch Kostenfunktionen ausgewählt.

Local Distcriminant Bases: ist eine Methode den Teilbaum zur Repräsentierung zu verwenden, der bei einem Klassifizierungsproblem die diskriminativsten Features ausweist. Anwendung: Texturklassifikation.

A trous Algorithmus Shift Stabilität wird erreicht durch Vermeidung des Downsampling, allerdings grosse Datenmenge (jeder Zerlegungsschritt führt zu gleicher Datenmenge wie Originalbild, siehe Fig. 85). Im Gegensatz zur CWT ist der DWT-Algorithmus einsetzbar. Allerdings wird die Skalierung recht grob gesampled (Oktavschritte).

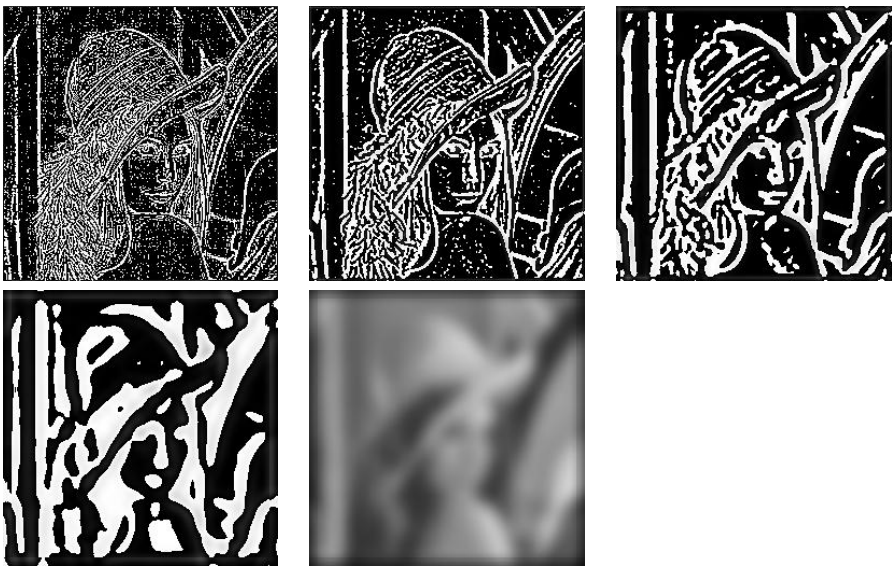


Figure 85: A trous Visualisierung

Stetige Wavelet Transformation (CWT) Direktes Berechnen der Koeffizienten mit $O(N^2)$ Komplexität. Hier wird ein explizit gegebenes Wavelet benötigt. Wird wegen enormer Datenmenge und grosser Komplexität meist nur bei 1-D Daten angewendet.

4 Image Restauration

Image Restauration sind Methoden zur Verbesserung der Bildqualität, wenn eine Bildstörung vorliegt, die beseitigt werden soll. Im Gegensatz zum Image Enhancement gibt es also ein Originalbild das möglichst gut wiederhergestellt werden soll. Die Art der Bildstörung ist entweder bekannt ist oder soll geschätzt

werden. Gründe für bekannte Störungen: Fehlfokussion, Bewegungsunschärfe, Rauschen (Übertragung, Sensorfehler, ...), Fehler im Linsensystem (Hubble), u.s.w.

deterministische Methoden für Bilder mit geringem Rauschen und bekannter Störfunktion

stochastische Methoden suchen die beste Restaurierung in Abhängigkeit von einem bestimmten statistischen/stochastischen Kriterium (least-squares criterion)

Je besser die Störung bekannt ist, desto besser ist die Restaurierung. Meistens muß die Störung allerdings geschätzt werden:

a priori Schätzung Störung ist bekannt oder wird vor Restaurierung erhalten

a posteriori Schätzung Bildanalyse anhand von *interessanten* Punkten (z.B. Kanten, gerade Linien) und man versucht zu rekonstruieren, wie diese ursprünglich waren.

4.1 Bildstörung

In dem im folgenden verwendeten Modell setzen wir eine positionsinvariante lineare Störung h und unabhängiges additives Rauschen vorraus:

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

$v(x,y)$ ... Rauschen
h ... Störung (meist positionsinvariant)

Aus der Linearität und dem Faltungssatz kann die Störung im DFT Bereich wie folgt dargestellt werden:

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

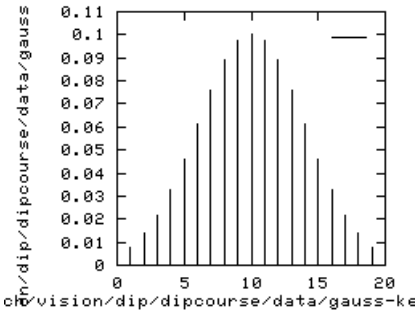
4.2 Bestimmung der Störung

Es gibt mehrere Möglichkeiten wie eine Bildstörung bestimmt (geschätzt, approximiert) werden kann. Da nie eine exakte Bestimmung gelingt, wird auch von "blind deconvolution" gesprochen.

4.2.1 Bildanalyse

Im gestörten Bild werden Regionen mit klarem Bildinhalt gesucht, z.B. eine scharfe Kante. Der entsprechende Bildausschnitt sei $g_s(x,y)$. Für diese Bildregion wird eine Approximation $f_s^a(x,y)$ erstellt. Da durch die Wahl des Ausschnitts Rauschen wenig Rolle spielen soll kann im DFT Bereich die DFT der Störfunktion im lokalen Bereich bestimmt werden:

Geglättetes Bild



Gauss Glättung (1-D)

Figure 86: Beispiel: Glättung als Bildstörung

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

Da Positionsinvarianz vorrausgesetzt wird, kann die Form der Störung auf das gesamte Bild übertragen werden.

4.2.2 Experimentelle Störungsbestimmung

Ist die Ausrüstung mit der das gestörte Bild aufgenommen wurde verfügbar (oder ein ähnlicher Typ), kann die Störung relativ genau abgeschätzt werden. Man nehme ein dem zu restaurierenden Bild ähnliches Bild und man versucht durch systematisches Testen der Systemkonfigurationen (z.B. Kameraeinstellungen) eine möglichst ähnliche Störung zu generieren. Dann wird ein kleiner starker Lichtpunkt aufgenommen, um die Impulsantwort der Störung zu erhalten (eine Störung des betrachteten Typs wird so vollständig charakterisiert). Die DFT eines Impulses ist eine Konstante A, so folgt:

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

4.2.3 Störungsbestimmung durch Modellierung

Wissen über Modelle von physikalischen Vorgängen wird benutzt.

Beispiele für einfache Störungen sind ...

relative (gleichmässige) Bewegung zwischen Kamera und Objekt

$f(x,y)$ bewegt sich so dass $x_0(t)$ und $y_0(t)$ die zeitabhängigen Bewegungskomponenten in x und y Richtung sind. Die gesamte Belichtung wird durch Integration über den Gesamtzeitraum der "Verschlussöffnung" T erreicht:

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

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

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

Die Reihenfolge der Integration kann vertauscht werden:

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

Der Ausdruck innerhalb der eckigen Klammern ist die DFT der verschobenen Funktion $f(x - x_0(t), y - y_0(t))$. Aus dem besprochenen Translationseigenschaften und der Unabhängigkeit zwischen $\hat{f}(u,v)$ und t ergibt sich

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

Folglich setzt man $\hat{h}(u,v) = \int_0^T e^{-2\pi i(ux_0(t)+vy_0(t))}dt$. Setzt man nun beispielsweise $x_0(t) = at/T$ und $y_0(t) = 0$ erhält man Bewegung nur in x-Richtung (BSP.: Aufnahme aus fahrendem Auto). Zum Zeitpunkt $t = T$ hat sich das Bild um die Distanz a bewegt. Wir erhalten

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

Für zweidimensionale Bewegung (also auch $y_0(t) = bt/T$) erhalten wir:

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

Fehlfokussierung

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

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

J_1 ... Besselfunktion erster Ordnung
a ... Ausmaß der Fehlfokussierung

Atmosphärische Turbulenz

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

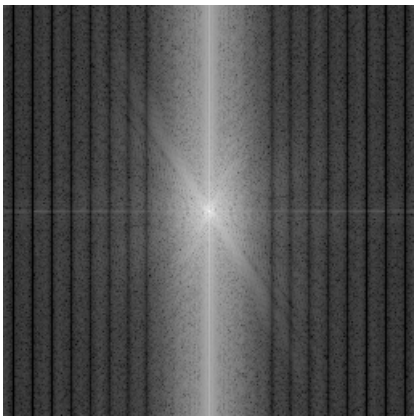
c wird experimentell ermittelt

4.3 Störungsbeseitigung

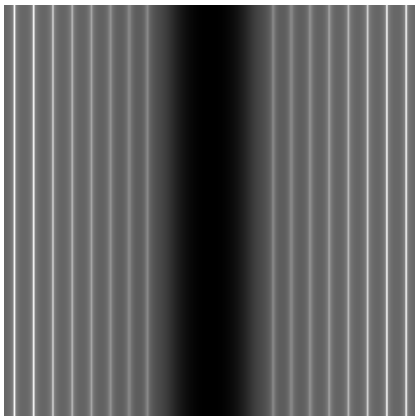
Um die Störung zu beseitigen braucht man also einen Restaurierungsfilter, der eine zur Störung inverse Transferfunktion $\hat{h}^{-1}(u,v)$ hat. Dies wird als "Inverse Filterung" bezeichnet.

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

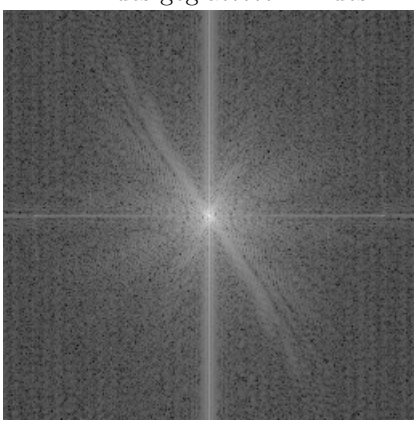
Ist der Rauschanteil nicht zu hoch, entspricht die Restaurierung einer inversen Faltung.



DFT des geglätteten Bildes



DFT des Gauss Kernels



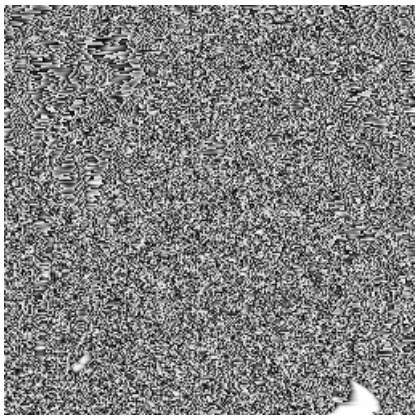
DFT nach inverser Filterung

Restauriertes Bild

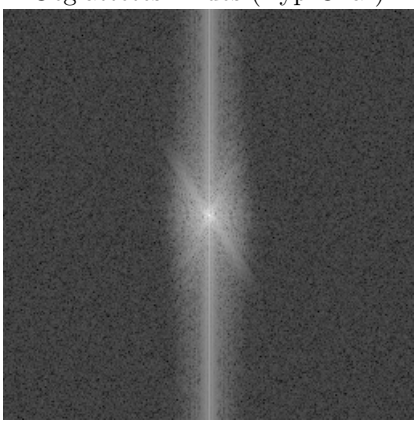
Figure 87: Beispiel Inverse Filterung

Wenn das Rauschen zu groß ist oder $\hat{h}(u,v)$ zu klein wird der Ausdruck $\hat{v}(u,v) \cdot \hat{h}^{-1}(u,v)$ zum Problem. Fig. 88 zeigt das Ergebnis der inversen Filterung mit hohem Rauschanteil: hier wurde das geglättete Bild (ursprünglich als `float` Datentyp) in `char` umgewandelt, was zu erheblichem Rauschanteil führt. Die Rekonstruktion ist entsprechend schlecht.

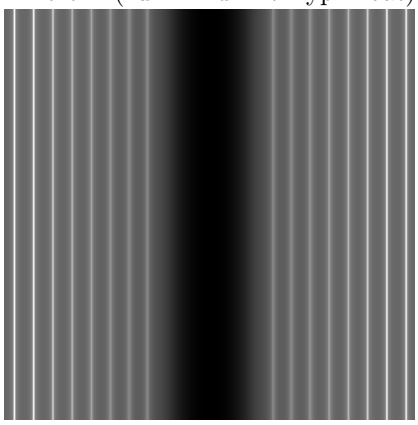
Geglättetes Bildes (Typ Char)



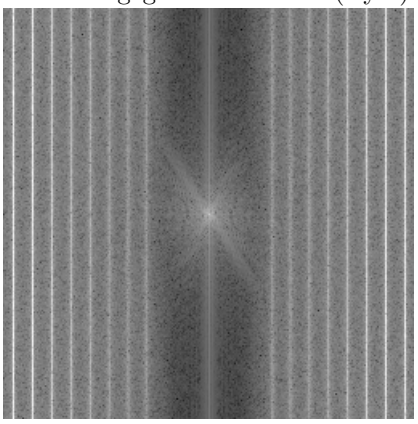
Differenz (zum Bild mit Typ Float)



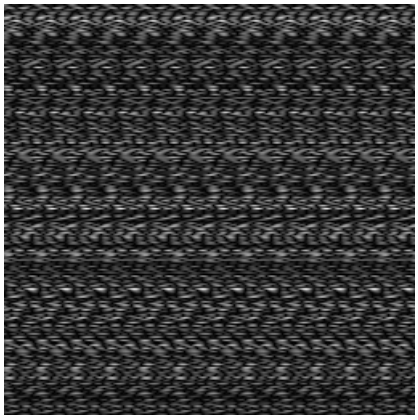
DFT des geglätteten Bildes (Byte)



DFT des Gauss Kernels



DFT nach inverser Filterung



Rekonstruiertes Bild

Figure 88: Beispiel Inverse Filterung mit Rauschen

Dies führt zur sog. “Pseudoinversen Filterung”:

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

Hier wird offensichtlich der Fall von zu kleinem $\hat{h}(u,v)$ abgefangen um ein Grosswerden des Ausdrucks $\hat{v}(u,v) \cdot \hat{h}^{-1}(u,v)$ zu verhindern. Grosse Werte für $\hat{v}(u,v)$ bleiben allerdings ein Problem.

DFT nach pseudo-inverser Filterung | Restauriertes Bild

Figure 89: Beispiel Pseudo Inverse Filterung

4.4 Wiener Filterung

Die Wiener Filterung nutzt nun zusätzlich a priori Wissen über das Rauschen aus. Restaurierung dieser Art gibt eine Abschätzung des ungestörten Bildes $\bar{f}$ mit minimalem Fehler $f(i,j) - \bar{f}(i,j)$ nach bestimmter Metrik. s_{xx} und $s_{\eta\eta}$ sind die Spektraldichten des Rauschens und des nicht gestörten Bildes (schwierig !).

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

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

Um die Wiener Filterung durchführen zu können braucht man Informationen über die Art der Störung und statistische Aussagen über das Rauschen. Ein Beispiel der Wiener Filterung ist in figure 90 zu sehen.

Da detailliertes Wissen zur Berechnung der Spekraldichten schwer zu gewinnen sein kann, wird auch ein sog. “parametrisierter” Wiener Filter verwendet:

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

Figure 90: Wiener Filterung

Dabei wird K optimiert bis das beste Ergebnis erreicht wird. Dies kann noch weiter verbessert werden was sog. "constrained least square" Filtern führt. Anschaulich bedeutet das, dass die Optimierung von K bezüglich eines least square Kriteriums durchgeführt wird, z.B. zur Maximierung der Bildglattheit (bzw. Minimierung der Bildunruhe), ausgedrückt durch einen Gradientenoperator.

5 Kantenerkennung

Die Begriffe Kante (*edge*) und *crack edge* wurden bereits in Kapitel 2.3.3 besprochen.

Zur Erinnerung: Kanten sind Pixel, wo die Helligkeitsfunktion die Größe verändert. Crack edges sind ein Kantenkonstrukt zwischen Pixeln (siehe figure 47).

Im Allgemeinen gibt es drei verschiedene Typen von Gradientenoperatoren:

1. Operatoren, welche Ableitungen der Bildfunktion durch Differenzen annähern: $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
2. Operatoren, welche Nullstellen der 2-ten Ableitung der Bildfunktion: $\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$ verwenden
3. Operatoren, welche die Bildfunktion auf ein parametrisches Kantenmodell abbilden

5.1 Methoden mit 1. Ableitung

5.1.1 Roberts Operator

Der Roberts Operator (siehe figure 93) arbeitet auf einer 2×2 Nachbarschaft mit den folgenden zwei Faltungsmasken und berücksichtigt die Richtung der Kanten ebenfalls nicht.

Nachteil
Extrem hohe Sensitivität gegenüber Rauschen, da nur sehr wenige Pixel zur Approximation verwendet werden.

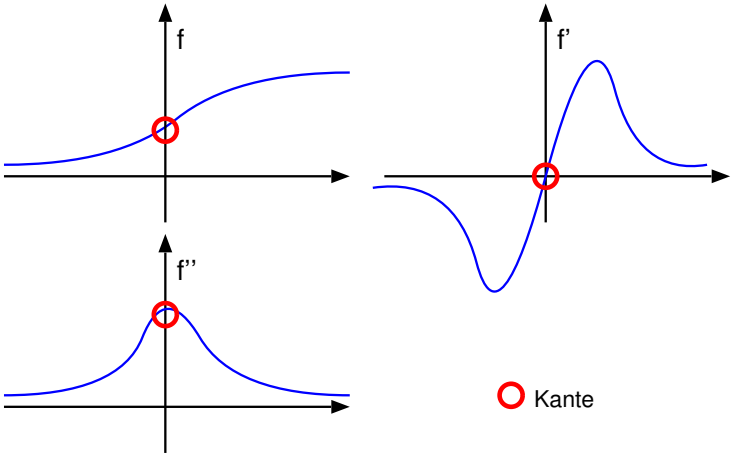


Figure 91: Visuell: 1. Ableitung vs. 2. Ableitung (wer findet die zwei Fehler in der Graphik ?)

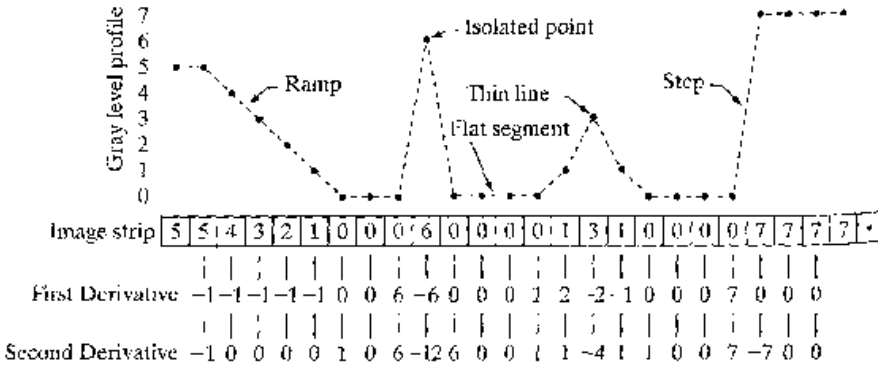


Figure 92: Numerik: 1. vs. 2. Ableitung

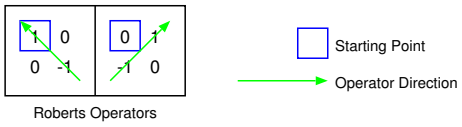


Figure 93: Roberts Operator

5.1.2 Kompass Operatoren

Die folgenden Operatoren werden auch *Kompass Operatoren* genannt, da sie die Richtung des Gradienten bestimmen. Der Gradient wird dabei in acht Richtungen berechnet und der größte Wert bestimmt die Richtung des Gradienten.

Prewitt Operator (siehe figure 94)

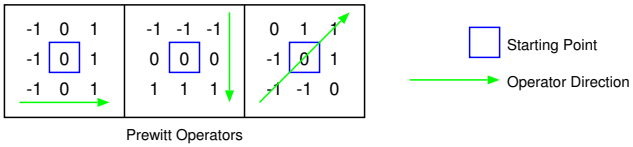


Figure 94: Prewitt Operator

Sobel Operator (siehe figure 95 und figure 96)

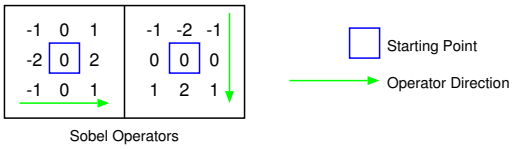


Figure 95: Sobel Operator

Figure 96: 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 97 ist die Anwendung von verschiedenen Kantenerkennungs-Operatoren dargestellt.

Nachteile

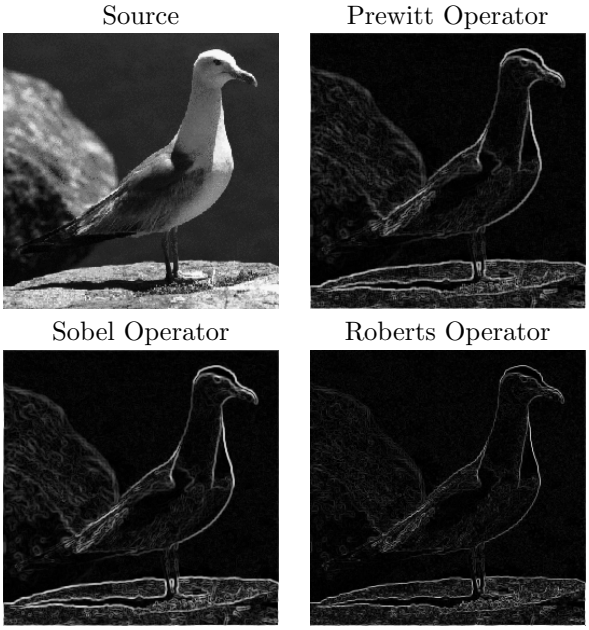


Figure 97: Edge Detection Examples

- Anfälligkeit gegen Rauschen
- Abhängigkeit von der Größe des Objekts bzw. von der Abruptheit der Kante. Und: es ist i.A. leichter Nullstellen zu finden als Extremwerte (siehe figure 92).

5.2 Methoden mit der 2. Ableitung

5.2.1 Laplace Operator

Ist man nur an Kantengrößen ohne Rücksicht auf Richtung interessiert, kann man den Laplace Operator verwenden, der rotationsinvariant ist (Erinnerung: Anwendung auch in der Fourier Domäne möglich).

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

Dieser wird oft durch 3×3 Masken für 4- und 8-Nachbarschaften angenähert

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

5.2.2 Mexican Hat Operator

Der Marr-Hildreth Operator (auch Mexican Hat Operator genannt) verwendet als zweidimensionalen Glättungsoperator einen Gauss'schen Filter

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

Die Standardabweichung σ ist proportional zur Größe der Nachbarschaft auf welcher der Filter operiert.

Um die zweite Ableitung zu berechnen (und die Nullstellen suchen zu können) wendet man den Laplaceoperator auf das geglättete Bild an.

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

$\bigtriangledown^2 G$ ist bildunabhängig und kann vorberechnet werden

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

Wenn man r^2 wieder durch $x^2 + y^2$ ersetzt so erhält man den *Laplacian of Gaussian* (LoG), welcher die Gestalt eines Mexican Hat (siehe figure 98) hat.

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

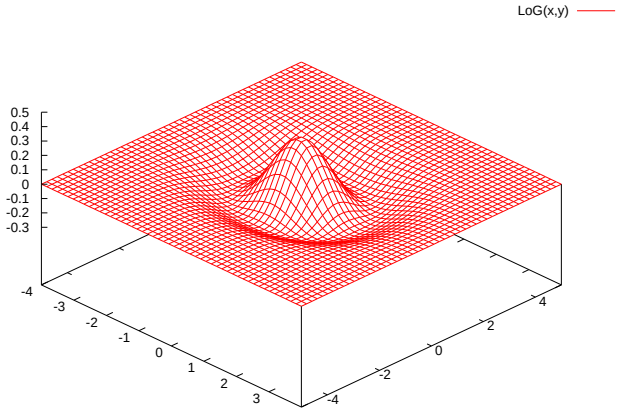


Figure 98: Mexican Hat

Fig. 99 zeigt einige Beispiele für ansteigende σ Werte, je grösser die Werte desto gröbere Kanten werden angezeigt (grosses σ in der Gauss Maske führt zu starker Glättung).

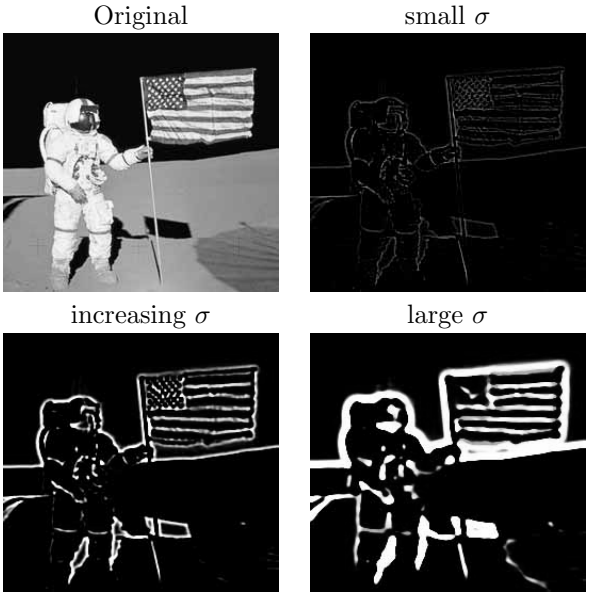


Figure 99: LoG Example

Bemerkung: $\bigtriangledown^2 G$ kann effizient durch die Differenz von zwei Gauß-Masken mit verschiedenem σ approximiert werden (*Difference of Gaussian*).

Vorteil
Kann mit σ den Maßstab bestimmen bzgl. dessen die Kanteneigenschaft definiert wird.

Nachteile

- glättet zum Teil sehr stark
- Trend zur Bildung von geschlossenen Kanten (*plate of sphagetti*)

5.3 Canny Edge Detector

Der Canny Edge Detector (für ein Beispiel siehe figure 101) wurde 1986 entwickelt und ist bezüglich der folgenden drei Kriterien optimal für verrauschte *step edges*

Detection keine wichtigen Kanten werden verfehlt und keine falschen angegeben

Localisation Abstand zwischen den tatsächlichen und den berechneten Kanten ist minimal

One response minimiert Mehrfachantworten auf eine Kante

Der Canny Edge Detector bietet folgende Features

thresholding with hysteresis verbessert die Erkennungsleistung wenn Rauschen

vorhanden ist. Dabei müssen Kanten den folgenden beiden Bedingungen gehorchen:

- Kantenwert > *high threshold*
- Kantenwert > *low threshold* und besteht Verbindung zu einer Kante > *high threshold*

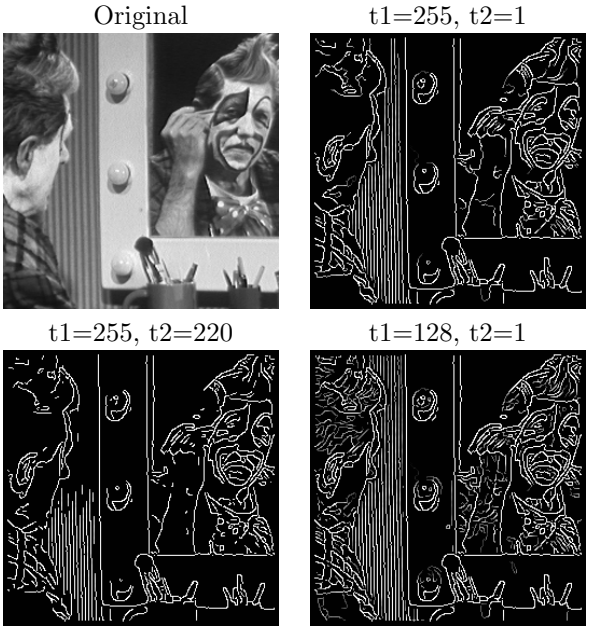


Figure 100: Edge Detection Examples: thresholding

non-maximal suppression nur lokale Maxima im Kantenbild werden verarbeitet

feature synthesis approach alle Kanten bezüglich eines kleinen Maßstabs (d.h. feine Kanten, kleines σ) werden markiert. Ein Prediktor für größere σ (siehe auch Kapitel 5.2.2) sagt Werte eines Operators mit größerem σ voraus: Glättung der feinen Kanten. Beim Vergleich mit den tatsächlich berechneten Werten werden nur Werte, die signifikant größer sind zusätzlich aufgenommen. Der gesamte Vorgang wird für mehrere σ durchgeführt.

Algorithmus

1. wiederhole 2 bis 5 für steigende Werte von σ
2. falte Bild mit Gaussian mit σ
3. berechne Gradientengröße und -richtung
4. finde Position der Kanten (non-maximal suppression)
5. thresholding with hysteresis
6. feature synthesis approach

Figure 101: Canny Edge Detector: verschiedene σ

Für ein Demo: `http://www.ii.metu.edu.tr/~ion528/demo/lectures/6/4/` und `http://www.cs.washington.edu/research/imagedatabase/demo/edge/`.

5.4 Line Finding Alogrithmen

5.4.1 Einfache Kernel

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

Die Gerade sei ein bis zwei Pixel breit, so kann ein Geraden-Bild mittels folgender Funktion und Maske erstellt werden

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

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

5.4.2 Hough Transformation

Eine **Gerade** ist definiert durch zwei Punkte $A = (x_1, y_1)$ und $B = (x_2, y_2)$. Alle Geraden durch A sind gegeben durch $y_1 = mx_1 + c$, für beliebige m und c (diese Gleichung gehört den Parameterraum m, c an). Alle Geraden durch A werden dargestellt als $c = -x_1m+y_1$ die durch B als $c = -x_2m+y_2$. Der einzige gemeinsame Punkt der beiden Geraden im m, c Parameterraum ist der Punkt, der die Gerade von A nach B darstellt. Jede Gerade im Bild wird dargestellt durch einen einzelnen Punkt im m, c Parameterraum.

Die Punkte (1,3), (2,2) und (4,0) liegen auf der gesuchten Geraden, (4,3) nicht.

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

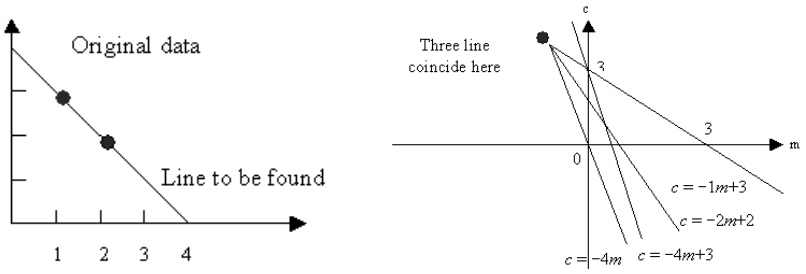


Figure 102: Grundprinzip der Houghtransformation

Aus der obigen Definition einer Geraden ergibt sich die folgende Vorgehensweise:

1. alle Geradenpixel bestimmen (durch Kantenerkennung)
2. alle Geraden die durch diese Punkte gehen bestimmen
3. die Geraden in den (m, c)-Parameterraum transformieren
4. die Punkte (a, b) im Parameterraum bestimmen, die aus der Houghtransform der Linie $y = ax + b$ entstanden sind

Bemerkung: Es wird nur eine beschränkte Anzahl von möglichen Geraden zugelassen. Somit entsteht ein *Akkumulatorarray*, dessen Elemente heißen *Akkumulatorzellen*.

Für jedes Kantenpixel werden Parameter m und c bestimmt, die in "erlaubte" Richtungen zeigen. Für jede mögliche Gerade wird m und c bestimmt und der Wert der dazugehörigen Akkumulatorzelle $A(m, c)$ erhöht. Geraden sind dann lokale Maxima des Akkumulatorarrays.

Vorteil
Robust gegen Rauschen

Vor- bzw. Nachteil
Interpoliert fehlende Geradenstücke

In der Praxis ist die Geradengleichung $y = mx + c$ für $m \to \infty$ ungünstig. In diesem Fall ist die Geradengleichung $r = x\cos(\theta) + y\sin(\theta)$ besser. Hier gibt es eine nette Visualierung:
`http://www.vision.ee.ethz.ch/~buc/brechbuehler/hough.html`

Allgemein wird eine Kurvendarstellung durch $f(x, a) = 0$ mit dem Vektor a der Kurvenparameter verwendet.

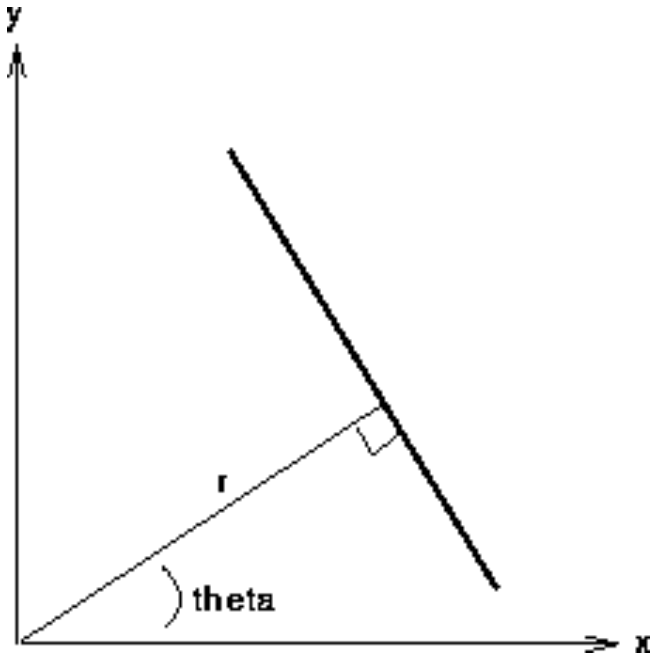


Figure 103: Alternative Geradendarstellung

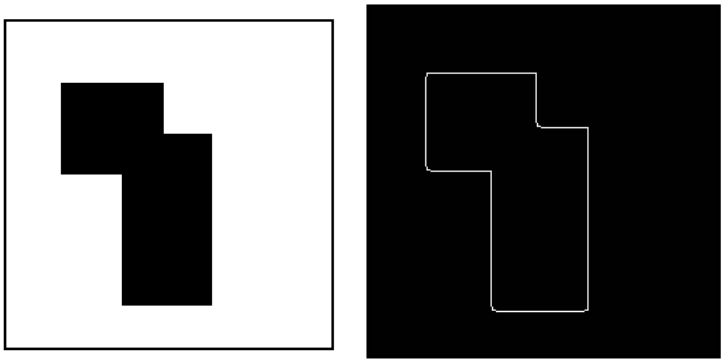


Figure 104: Beispiel der Houghtransformation

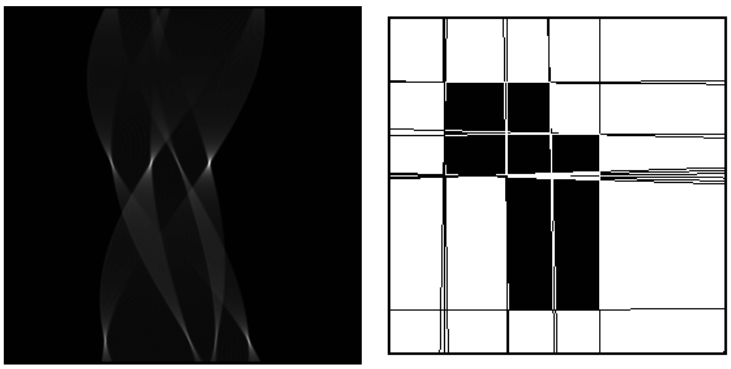


Figure 105: Beispiel der Houghtransformation

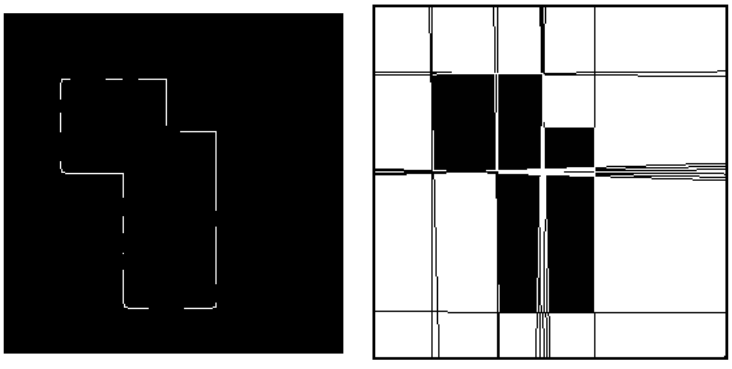


Figure 106: Beispiel der Houghtransformation

Algorithmus

1. Quantisierung des Parameterraums, Dimension n des Raumes ist Anzahl der Parameter in q
2. bilde n-dimensionale Akkumulatorarray $A(a)$ und setze $A(a) = 0$
3. $\forall(x_1, y_1)$ im geeignet ge-thresholdeten Gradientenbild erhöhe die Zelle $A(a)$ wenn $f(x, a) = 0$
4. lokale Maxima im Akkumulatorarray sind Geraden

Bei Kreisen $(x_1 - a)^2 + (y_1 - b)^2 = r^2$ werden drei Parameter und ein dreidimensionaler Akkumulatorarray benötigt. Für kompliziertere Kurven werden hochdimensionale Akkumulatorarrays verwendet.

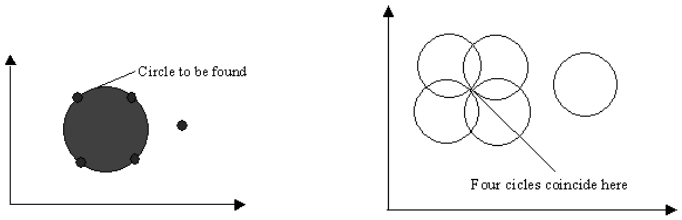


Figure 107: Zirkuläre Houghtransformation

6 Image Segmentation

Image Segmentation ist einer der wichtigsten Schritte in der Bildanalyse. Hauptziel ist es ein Bild in Teile aufzuteilen, die eine hohe Korrelation mit Objekten oder Gebieten der echten Welt haben.

complete segmentation eine Menge durchschnittsfreier Regionen/Objekte, die eindeutig realen Objekten entsprechen. Oft wird dafür *Expertenwissen* im Zusammenhang mit KI Methoden benötigt. Manche Applikationen sind allerdings einfach genug, z.B. Schrift, Objekte vor gleichmässigem Hintergrund und hohem Kontrast im Allgemeinen.

partial segmentation : hier werden Regionen erkannt die homogen bzgl. eines bestimmten Kriteriums sind, die aber nicht unbedingt mit den Objekten der Szene übereinstimmen.

Es gibt drei Gruppen von Segmentierungsverfahren:

1. Verfahren, die globales Wissen über das Bild oder seine Teile verwenden (z.B. Histogramm)
2. Kanten-basierte Verfahren versuchen geschlossene Kantenzüge als Objektgrenzen zu erhalten

3. Region-basierte Verfahren (die beiden letzten Verfahren lösen ein duales Problem: jede Region kann durch ihre geschlossene Grenzkurve beschrieben werden und jede geschlossene Kurve beschreibt eine Region).

6.1 Thresholding

Thresholding ist das einfachste SegmentierungsVerfahren, aber wegen hoher Geschwindigkeit immer noch aktuell. Eine Helligkeitskonstante oder Schranke (Threshold) T wird bestimmt um Objekte vom Hintergrund zu trennen. Somit wird das Inputbild in ein binäres, segmentiertes Outputbild transformiert:

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

Thresholding ist also das passende Verfahren, wenn sich Objekte nicht berühren und deren Grauwerte vom Hintergrund verschieden sind ist.

Bei Thresholding ist besonders wichtig die Wahl des Thresholds! Selten ist Thresholding mit einem fixen Threshold erfolgreich. Besser ist variables Thresholding ($T = T(f, f_c)$ mit f_c für den betrachteten Bildteil) das den Schrankenwert als Funktion variabler Bildvharakteristik variiert.

6.2 Thresholding Variationen

Band Thresholding man sucht Grauwert aus bestimmtem Grauwert-Bereich (z.B. mikroskopische Bakterien-Zellen Segmentierung, bekannte grauwerte durch bekannte Materialdichte bei CT-Bildern)

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

Multi Thresholding verwendet mehrere Schranken und das Ergebnisbild ist nicht binär.

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

Semi Thresholding Ziel ist einen eventuell vorhandenen Hintergrund zu entfernen und die Grauwert-Information in den Objekten zu erhalten

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

6.3 Wahl des Thresholds

Bei einem Text weiß man, daß Buchstaben einen bestimmment Anteil des Bildes ($1/p$ der Blattfläche) bedecken. Die Schranke kann durch Histogrammanalyse

leicht gewählt werden, sodaß $1/p$ des Bildes $\leq T$ ist (heisst "p-tile thresholding").

Schrankenbestimmung muß meistens durch Histogrammanalyse erfolgen da man solche Informationen meist nicht hat.

- Objekte mit gleichem Grauwert der sich vom Grauwert des Hintergrunds unterscheidet: das Histogramm ist *bimodal*. Der Threshold wird als minimaler Wert zwischen den Extrema gewählt.
- Bei *multimodalen* Histogrammen können oder müssen mehrere Schranken zwischen je zwei Maxima gewählt werden, es entstehen verschiedene Segmentierungsergebnisse oder man nimmt Multithresholding.

Problem
Wie entscheidet man ob das Histogramm bimodal oder multimodal ist? Bimodale Verfahren suchen normal die beiden höchsten Maxima und setzen T als Miniumum dazwischen ("mode method"). Um zu vermeiden, dass zwei nahe beisammen liegende lokale Maxima gewählt werden, sollte ein minimaler Grauwert-Abstand von Maxima verlangt werden, oder es wird eine Histogrammglättung durchgeführt. Achtung: ein bimodales Histogramm garantiert noch keine korrekte Segmentierung, z.B. 50% S/W Pixel vermischt und auf einer Seite konzentriert haben identisches bimodales Histogramm.

Weitere Verfahren zur Threshold Wahl

lokale Nachbarschaften Berücksichtigung von lokalen Nachbarschaften bei der Erzeugung des Histogramms (z.B. durch Gewichtung von Pixeln, z.B. um Pixel mit hohem Gradienten zu unterdrücken; dann besteht das Histogramm nur aus Objekt- und Hintergrundpixeln, die Grenzpixel werden unterdrückt)

Optimal Thresholding Histogramm wird durch eine gewichtete Summe von Wahrscheinlichkeitsdichten approximiert. Die Schranke wird gleich der minimalen Wahrscheinlichkeit zwischen den Maxima der Verteilungen gesetzt.

Problem: Schätzung der Verteilungsparameter und erkennen der Art der Verteilung (meist Normalverteilung) ist aufwendig.

Iterative Threshold Wahl Man nimmt an, daß es Regionen mit zwei Hauptgrauwerten gibt. Die vier Ecken des Bildes enthalten Hintergrundpixel.

In der t-ten Iteration berechne die Mittelwerte μ_B^t (Background) und μ_O^t (Object) wobei die Segmentierung im t-ten Schritt definiert ist durch

$$T^t = \frac{\mu_B^{t-1} + \mu_O^{t-1}}{2}$$

$$\mu_B^t = \frac{\sum_B f(i,j)}{\text{Anzahl Backgroundpixel}}$$

$$\mu_O^t = \frac{\sum_O f(i,j)}{\text{Anzahl Objectpixel}} T^{t+1} = \frac{\mu_B^t + \mu_O^t}{2}$$

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

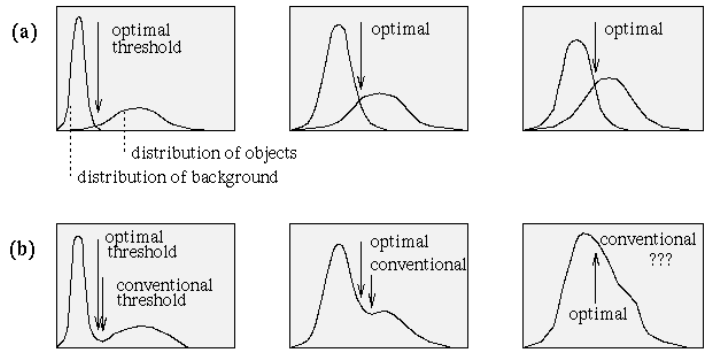


Figure 108: Konzept von optimalen Thresholding

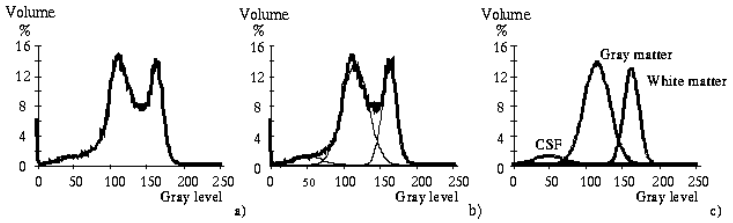


Figure 109: Beispiel für optimales Thresholding: Histogramm

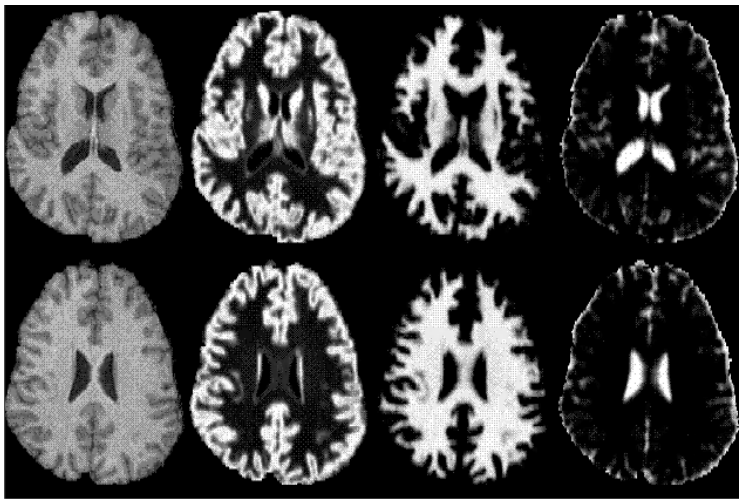


Figure 110: Beispiel für optimales Thresholding: Ergebnis

Thresholding in hierarchischen Datenstrukturen verwendet Pyramidenstruktur (siehe auch Kapitel 2.6). Man versucht im Bild mit geringer Auflösung Regionen zu entdecken und gibt ihnen in höherer Auflösung mehr Präzision. Zwei mögliche Varianten:

1. Segmentierung im low-resolution Bild mittels Thresholding; in der nächst höheren Auflösung werden Pixel nahe den Grenzen neu zugeordnet, dies wird sukzessive bis zur höchsten Auflösung durchgeführt.
2. Ein *significant pixel detector* sucht im Bild mit der niedrigsten Auflösung Pixel die sich von ihrer Umgebung unterscheiden, dies wird meist mit 3 x 3 Masken untersucht, die einen von der Umgebung abweichenden Zentralpixel anzeigen. Solche Pixel sollen in hoher Auflösung eigenen Regionen entsprechen. Der entsprechnede Teil des Bildes in voller Auflösung wird ge-thresholdet, T wird gesetzt zwischen dem Grauwert des signifikanten Pixel und dem Durchschnitt der restlichen 8-Nachbarn (lokal mit verschiedenen Thresholds für verschiedene Regionen)

Vorteil der hierarchischen Verfahren ist die höhere Robustheit gegenüber Rauschen, da die Initialsegmentierungen von einem stark geglätteten Bild ausgehen.

6.4 Kantenbasierte Verfahren

Kanten-basierte Verfahren zur Segmentierung sind die ältesten Verfahren. Nach der Kantenerkennung ist das Bild noch nicht segmentiert; es müssen erst die Kantenpixel zu Kantenketten (*edge chains*) kombiniert werden, die besser Regionengrenzen entsprechen. Die folgenden drei Unterkapitel widmen sich dieser Art der Segmentierung.

6.4.1 Kantenbild Thresholding

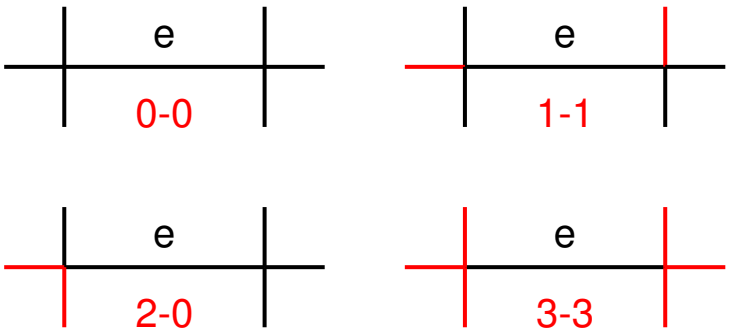
Kleine Werte im Kantenbild entsprechen nicht signifikanten Grauwert-Wechseln; diese Werte können durch einfaches Thresholding ausgeschlossen werden. Eine einfache Weiterverarbeitung wäre es isolierte Kantenpixel oder Segmente unterhalb einer bestimmter Längenschranke auszuschließen.

6.4.2 Edge Relaxation

Kanten Thresholding wird durch Rauschen stark beeinträchtigt, oft fehlen Kantenteile um vollständige Regionengrenzen zu bilden. Bei Edge Realxation wird die Kanteneigenschaft wird im Kontext der Nachbarpixel betrachtet (vgl. Thresholding with hysteresis). Unter Berücksichtigung der Kantenstärke und Kantenfortsetzung wird die Stärke der Kanteneigenschaft iterativ verstärkt oder abgeschwächt. Hier werden Kanten als *crack edges* (siehe auch Kapitel 2.3.3) interpretiert.

Kanteneigenschaft wird an beiden Enden einer Kante betrachtet mit allen drei möglichen Positionen. Kante e hat eine Kreuzung an jedem Ende und es

gibt je drei mögliche Weiterführungen (siehe figure 111). Jede Kreuzung wird entsprechend ihrer wegführenden Kantenanzahl und -form bewertet. Die Kanteneigenschaft $c^1(e)$ in der ersten Iteration ist die normalisierte Größe der crack edge. Diese konvergiert dann von Iteration zu Iteration gegen 0 oder 1.



0-0	isolierte Kante	-	0-2	dead end	-
0-3	dead end	-			
0-1	neutral	0	2-2	Brücke	0
2-3	Brücke	0	3-3	Brücke	0
1-2	Weiterführung	+	1-3	Weiterführung	+
1-1	Weiterführung	++			

Figure 111: Kanteneigenschaften

Algorithmus (2. und 3. werden iteriert)

1. evaluiere Kanteneigenschaft $c^1(e)$ für alle crack edges im Bild
2. finde die Kantentypen in der Nachbarschaft und die Arten der Kreuzungen
3. update von $c^{k+1}(e)$ für jede Kante entsprechend ihres Typs und $c^k(e)$
4. Abbruchkriterium (z.B. wenn Kanteneigenschaften gegen 0 oder 1 konvergieren)

Bewertung der Kreuzungstypen
Kreuzung ist vom Typ i, wenn $\text{type}(i) = \max_k((\text{type}(k)), k = 0, 1, 2, 3$

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

a, b, c ... normierte Werte der anliegenden Kanten
m ... $m = \max(a, b, c, q)$
q ... konstant; $q \sim 0.1$

Beispiel: $(a, b, c) = (0.5, 0.05, 0.05)$ ist eine Typ 1 Kreuzung, $(0.3, 0.2, 0.2)$ eine von Typ 3.

Ähnliche Ergebnisse werden durch Zählen der Anzahl der Kanten an einer Kreuzung über einer Schranke erhalten.

Update Schritt

Kanteneigenschaft verstärken : $c^{k+1}(e) = min(1, c^k(e) + \delta)$

Kanteneigenschaft abschwächen : $c^{k+1}(e) = max(0, c^k(e) - \delta)$

δ wird typischerweise im Bereich 0.1 - 0.3 gewählt (für starke und schwache Modifikation), einacher mit einem Wert. In der bisherigen Form wird oft nur ungenügend langsame Konvergenz erreicht.

Verbesserter Update Schritt

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

6.4.3 Kantenketten als Graphsuche

Vorwissen: Anfang und Endpunkt eines Kantenzugs
Graph: Menge von Kanten n_i und Pfaden $[n_i, n_j]$ zwischen diesen Kanten

Wir betrachten orientierte und gewichtete Pfade wobei die Gewichte als Kosten bezeichnet werden.

Die Kantensuche wird in eine Suche nach einem optimalen Pfad im gewichteten Graphen transformiert. Gesucht ist der beste Pfad zwischen Anfangs und Endpunkt. Es sei Information über Kantengröße $s(x)$ und Kantenrichtung $\phi(x)$ vorhanden. Jedes Pixel entspricht einem mit $s(x)$ gewichtet Knoten. Zwei Kanten n_i und n_j sind durch einen Pfad verbunden, wenn $\phi(x_i)$ und $\phi(x_j)$ zusammen passen: x_i muß einer der drei existierenden Nachbarn von x_j in der Richtung $d \in [\phi(x_i) - \pi/4, \phi(x_j) + \pi/4]$ sein und $s(x_i)$ und $s(x_j) \geq T$.

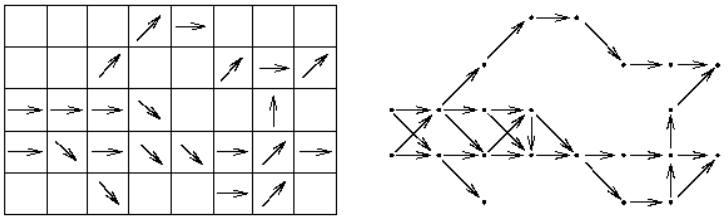


Figure 112: Graphen Repräsentierung eines Kantenbildes

Nun können Graphen-Suchmethoden angewendet werden. Seien x_A und x_B Start- und Endpunkt. Es muß eine Methode zur Expansion und eine Kostenfunktion $f(x_i)$ definiert werden, die eine Kostenschätzung eines Pfades zwischen x_A und x_B durch x_i erlaubt. Die Kostenfunktion muß aufteilbar und monoton bezüglich der Pfadlänge sein.

$g(x_i)$	Kosten von x_A nach x_i Summe der Kosten der Knoten im Pfad von x_A nach x_i
$h(x_i)$	Kosten von x_i nach x_B (Schätzung)

Nielson's A-Algorithmus (heuristische Graphensuche)

1. expandiere x_A und gebe alle Nachfolger in eine OPEN-Liste mit Pointern zurück auf x_A; berechne Kostenfunktion für jeden Knoten
2. wenn OPEN-Liste leer ist, dann ist der Algorithmus fehlgeschlagen. sonst: bestimme x_i in der Liste mit den geringsten Kosten $f(x_i)$ und entferne diesen Knoten. Wenn $x_i = x_B$ folge den Pointern um den besten Pfad zu finden und stop.
3. war nicht stop in 2. expandiere x_i und gib die Nachfolger in die OPEN-Liste mit Pointern zurück auf x_i; berechne deren Kosten; gehe nach 2.

Wichtig bei dem Algorithmus ist ein Mechanismus gegen Schleifenbildung. Die Schätzung $\hat{h}(x_i)$ von $h(x_i)$ hat wesentlichen Einfluß auf das Suchverhalten (kann Suche beschleunigen - ungenau - oder zu vollständiger Suche machen).

Varianten

$\hat{h}(x_i) = 0$: keine Heuristik ist inkludiert und das Ergebnis ist eine breadth-first Suche. heuristische Methoden garantieren zwar nicht das optimale Ergebnis dafür sind sie schneller.

$\hat{h}(x_i) > h(x_i)$: der Algorithmus wird schnell laufen, aber ein Ergebnis mit minimalen Kosten kann nicht garantiert werden.

$\hat{h}(x_i) = h(x_i)$: die Suche findet den Pfad mit minimalen Kosten unter Verwendung einer minimalen Anzahl von exandierten Knoten. Allgemein gilt dass die Anzahl der expandierten Knoten umso kleiner ist, je näher $\hat{h}(x_i)$ an $h(x_i)$ liegt.

$\hat{h}(x_i) \leq h(x_i)$: die Suche liefert den Pfad mit minimalen Kosten, wenn auch für jeden Teilpfad gilt dass die wahren Kosten grösser sind als die geschätzten.

Branch and Bound Algorithmen maximal erlaubte Kosten werden definiert und teurere Pfade nicht weiter betrachtet

Multiresolution Processing Modelle werden zuerst im low-resolution-Bild angewendet

Mögliche Kostenfunktionen:

- Kantenstärke: hohe Kantengrösse → gute kante → kleine Kosten (z.B. direkte Kosten: Differenz zum grössten Kantenwert im Bild).
- Krümmung: Unterschied der Kantenrichtungen
- Abstand zum bekannten Endpunkt
- Anstand zur vermuteten oder bekannten Position der Kante

Dynamische Programmierung

Grundlage: Bellmann'sches Optimalitätsprinzip – unabhängig vom Pfad zum Knoten E, gibt es einen optimalen Pfad zwischen E und dem Endpunkt. Mit

anderen Worten: Geht der optimale Pfad zwischen Anfangs- und Endpunkt durch E, dann sind auch die Pfadteile Anfangspunkt → E und E → Endpunkt optimal.

1. Erstellung des Graphen und Bewertung aller Teilpfade zwischen je zwei Schichten.
2. In jeder Schicht wird für jeden Knoten E bestimmt, welcher der Teilpfad aus der vorherigen Schicht mit den geringsten Kosten nach E ist.
3. In der letzten Schicht wird der Knoten mit den geringsten Kosten bestimmt.
4. Durch Backtracking entlang der gefundenen Teilpfade wird der optimale Pfad bestimmt.

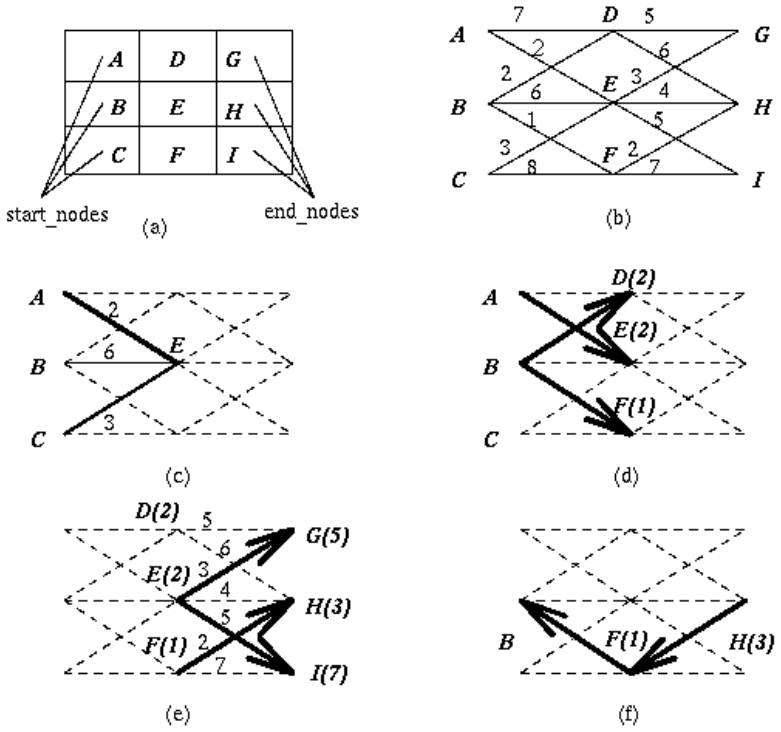


Figure 113: Beispiel Dynamic Programming: (a) Kantenbild (b) Graph mit Kosten (c) mögliche Pfade nach E, A-E ist optimal (d) optimale Pfade nach D,E,F (e) optimale Pfade nach G,H,I (f) Backtracking von H bestimmt Pfad mit geringsten Kosten

6.5 Regionsbasierte Verfahren

Regionsbasierte Verfahren werden vor allem in verrauschten Bildern angewendet, wo Kantenerkennung schlecht funktioniert. Dabei wird das Bild in Regionen

maximaler Homogenität aufgeteilt.

Homogenität
Die Homogenität wird meist Grauwert-basierend definiert (z.B. durchschnittlicher Grauwert, Form eines lokalen Histogramms) oder verwendet Textureigenschaften u.s.w.

Regionen haben folgende Eigenschaft (neben ihrer Durchschnittsfreiheit):

$$\begin{aligned} H(R_i) &= \text{True} && i = 1, \ldots, S \\ H(R_i \cup R_j) &= \text{False} && i \neq j \text{und} R_i \text{ benachbart zu } R_j \end{aligned}$$

$S \ldots$ Anzahl der Regionen, $H(R_i)$ ist eine binäre Homogenitätsevaluierung von R_i; d.h., Regionen sind homogen und maximal (wenn sie grösser wären, wären sie nicht mehr homogen).

In den folgenden zwei Kapiteln werden regionsbasierte Verfahren vorgestellt.

6.5.1 Region Merging

1. segmentiere Bild in kleine Regionen die dem Homogenitätskriterium genügen
2. definiere Kriterien um Regionen wieder zu vereinigen (*merging*)
3. führe merging durch bis es nicht fortgesetzt werden kann

Verschiedene Methoden unterscheiden sich durch verschiedene Startsegmentierungen und verschiedene Kriterien für Homogenität. Eine Verbesserung ist es, wenn zusätzlich Kanteninformation verwendet wird: benachbarte Regionen werden gemerged, wenn ein wesentlicher Teil (d.h. abhängig von der Gesamtlänge) ihrer gemeinsamen Grenze aus "schwachen" Kanten besteht. Kantensignifikanz ist z.B. der Output von edge relaxation.

6.5.2 Region Splitting

Umgekehrt zu *Region Merging* wird mit dem ganzem Bild begonnen das normalerweise nicht dem Homogenitätskriterium genügt. Das Bild wird in Regionen gesplittet, die dann dem Homogenitätskriteriums genügen.

Bemerkung: Merging ist nicht dual zu splitting! Selbst bei gleichem Homogenitätskriterium. Siehe dazu splitting und merging in figure 114 eines Schachbrett mit Gleichheit des durchschnittlichen Grauwerts als Homogenitätskriterium. Beim Splitting ist das Homogenitätskriterium für alle vier Quandranten Mittelgrau, beim merging schwarz oder weiss bis man zur Grösse des Schachbretts kommt.

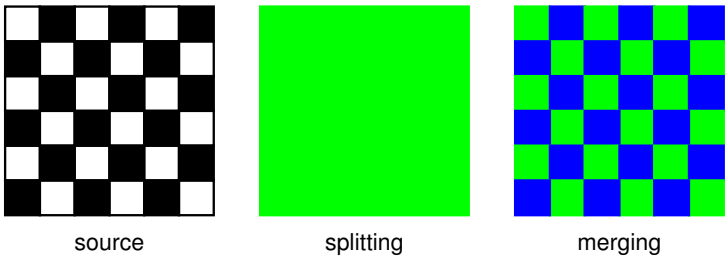


Figure 114: splitting/merging

Anwendung
Häufig wird eine Kombination von Splitting und anschließendem Merging in einer Baumstruktur – häufig Quadtrees – verwendet (*split and merge*):

1. Definiere eine initiale Segmentierung in Regionen, ein Homogenitätskriterium und eine pyramidale Datenstruktur.
2. Ist eine Region in der Datenstruktur nicht homogen, wird sie in ihre 4 Kinderregionen gesplittet; wenn 4 Regionen mit dem gleichen Elternknoten gemerged werden können, soll das durchgeführt werden. Wenn keine Region mehr bearbeitet werden kann, GOTO 3)
3. Gibt es zwei benachbarte Regionen entweder auf verschiedenen Pyramidenebenen oder mit unterschiedlichen Elternknoten die dem Homogenitätskriteriumentsprechen sollen sie gemerged werden.
4. Zu kleine Regionen sollen mit der ähnlichsten benachbarten Region gemerged werden.

6.5.3 Template Matching

Beim Template Matching werden bekannte Objekte durch Berechnung der Abweichung von *templates* lokalisiert.

6.6 Watershed Segmentierung

Dieser Segmentierungsansatz verwendet Methoden aus der morphologischen BVA und wird daher im folgenden Kapitel als abschliessende Anwendung angeführt.

7 Morphologische Bildverarbeitung

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 3×3 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 3×3 neighborhood in a binary image.

Convolution of a binary image with a 3×3 kernel (see figure 115) generates a nine-bit (512-gray-level) image in which the gray level of each pixel specifies the configuration of the 3×3 binary neighborhood centered on that point. Neighborhood operations thus can be implemented with a 512-entry look-up table with one-bit output.

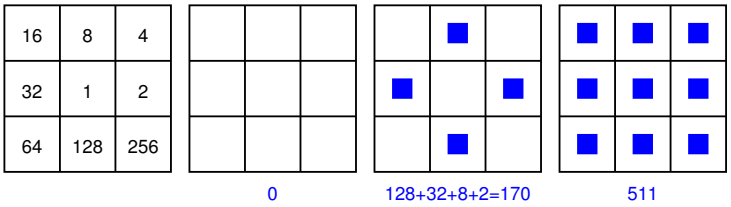


Figure 115: Binary neighborhood encoding

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 116). 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 upen the size and content of the structuring element and upon the nature of the logical

operation.

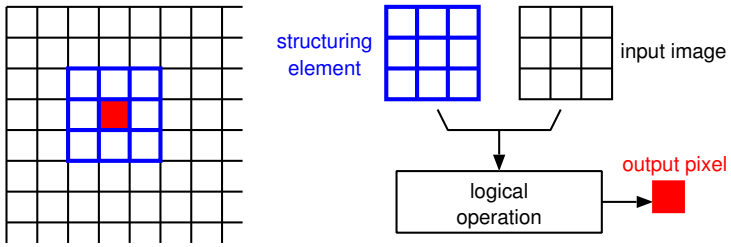


Figure 116: Morphological image processing

For this introduction to the subject, we concentrate on the simplest case, namely, the use of a basic 3×3 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, $\mathbf{B}$, and the structuring element, $\mathbf{S}$, are sets defined on a two-dimensional Cartesian grid, where the 1's are the elements of those sets.

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 117). By definition, a boundary point is a pixel that is located inside an object, but that has at least one neighbor outside the object.

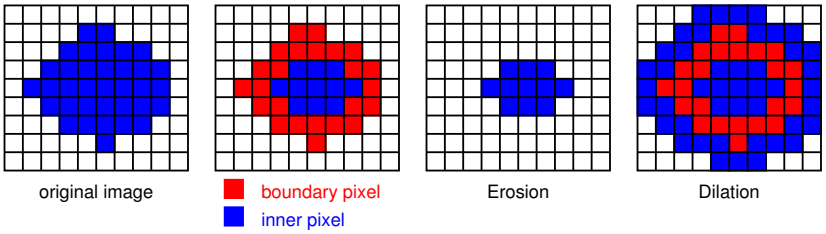


Figure 117: Erosion and dilation

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 3×3 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 $\mathbf{B}$ that results from dilating $\mathbf{B}$ 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 its intersection with $\mathbf{B}$ is not empty. With the basic 3×3 structuring element, this reduces to simple dilation.

7.1.3 Opening and Closing

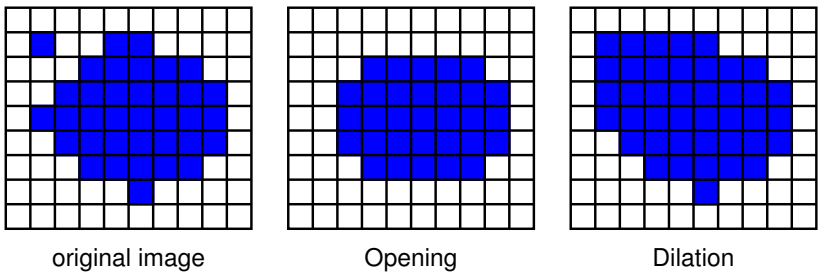


Figure 118: Opening and closing (auch in der Figure !)

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 3×3 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

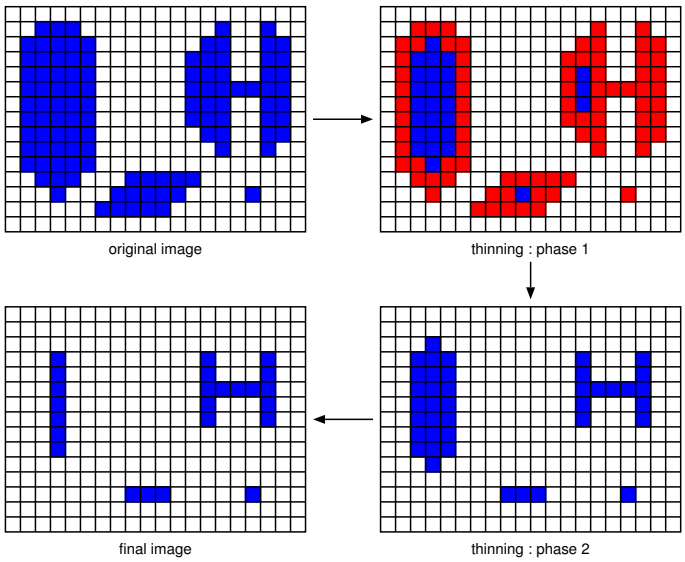


Figure 119: Thinning

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 3×3 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 119).

7.4 Skeletonization

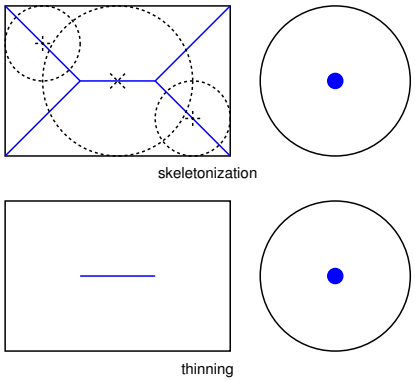


Figure 120: Skeletonization

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.

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 120).

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 3×3 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 Anwendung: Watershed Segmentierung

Für eine gute (animierte) Visualisierung und diverse Beispiele siehe:
`http://cmm.ensmp.fr/~beucher/wtshed.html`

Wasserscheiden trennen einzelne Auffangbecken – um diese Begriffe aus der Topographie auf die BVA übertragen zu können wird ein Bild als dreidimensional interpretiert: die Helligkeitswerte werden zu Höhenangaben (über den entsprechenden Koordinaten). Meistens wird dann ein Gradientenbild zur Weiterverarbeitung verwendet. Regionen Grenzen (i.e. Kantenketten) entsprechen "hohen" Wasserscheiden und innere Regionen mit niederem Gradienten entsprechen den Auffangbecken (siehe Fig. 121).

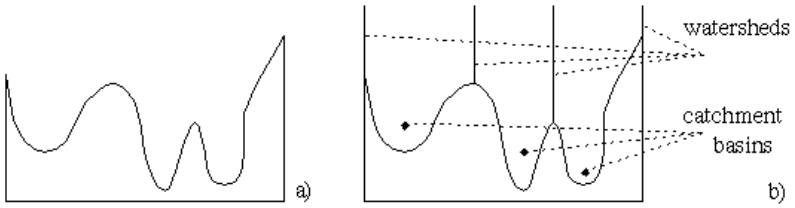


Figure 121: Grundprinzipien: Wasserscheiden und Auffangbecken

Auffangbecken sind homogen in dem Sinn dass alle Pixel die zum gleichen Auffangbecken gehören mit dem Minimum des Auffangbeckens durch einen Pfad verbunden sind, dessen Pixel monoton abnehmend in Richtung des Minimums sind. Die Auffangbecken repräsentieren die Regionen des segmentierten Bildes, die Wasserscheiden sind die Regionengrenzen.

Es gibt zwei grundlegende Ansätze für Watershed Segmentierung:

1. Der erste Ansatz such zuerst für jedes Pixel im Bild einen "downstream" Pfad zu einem Minimum. Ein Auffangbecken ist dann definiert als Menge aller Pixel deren Pfad zum gleichen Minimum führt. Das Problem dieses

Ansatzes ist die eindeutige Bestimmung des Pfades (die im stetigen Fall durch lokale Gradienten gewährleistet werden kann).

2. Der zweite Ansatz ist dual zum ersten und verwendet "flooding": die Auffangbecken werden von unten her geflutet (Annahme: an den Stellen lokaler Minima sind Löcher, bei Eintauchen der Oberfläche in Wasser wird durch die Löcher geflutet). Würden nun zwei Auffangbecken durch das steigende Wasser zusammenfallen, wird ein Damm errichtet der das verhindert, der Damm ist so hoch wie der grösste Wert im Bild.

Im folgenden betrachten wir das zweite Verfahren näher. Zur Vorverarbeitung werden werden die Pixel entsprechend ihres Grauwerts sortiert, das Grauwerthistogramm ermittelt und eine Liste von Pointern auf alle Pixel mit Grauwert h erstellt. So können alle Pixel mit einem bestimmten Grauwert angesprochen werden. Sei das Fluten fortgeschritten bis zum Grauwert k. Jedes Pixel mit Grauwert $\leq k$ wurde bereits eindeutig einem Auffangbecken zugeordnet und trägt dessen Label. Im nächsten Schritt werden alle Pixel mit Grauwert $k+1$ bearbeitet. Ein Pixel mit diesem Grauwert kann zum Auffangbecken l gehören, wenn zumindest ein direkter Nachbar dieses Label trägt. Um die Zugehörigkeit zu einem Auffangbecken zu bestimmen werden Einflusszonen bestimmt: die Einflusszone eines Auffangbeckens l sind die Positionen der nicht-zugeordneten mit dem Auffangbecken verbundenen Pixel mit Grauwert $k+1$, deren Abstand zu l kleiner ist als zu jedem anderen Auffangbecken. Siehe Fig. 122 für eine Visualisierung.

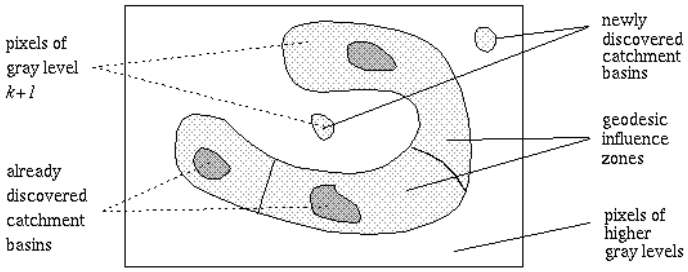


Figure 122: Einflusszonen der Auffangbecken

Alle Pixel mit Grauwert $k+1$ die zur Einflusszone des Auffangbeckens l gehören bekommen das Label l, d.h. die Auffangbecken wachsen. Die nicht-zugeordneten Pixel werden sukzessive abgearbeitet, Pixel die kein Label zugewiesen bekommen entsprechen neuen Auffangbecken und bekommen ein neues Label. Die Grenzen zwischen den so entstandenen Auffangbecken sind die Watersheds. Fig. 123 stellt den gesamten Vorgang dar.

Es ist zu beachten, dass bei dem eben beschriebenen Verfahren keine explizite Dammberechnung durchgeführt wird (dies wird im Anschluss mittels morphologischen Operationen erklärt) . Wird dieses Verfahren nun direkt angewendet, ist das Resultat häufig eine starke Übersegmentierung (d.h. zu viele Regionen, siehe Figs. 124 und 126.c), da es oft eine zu hohe Anzahl an Minima gibt (z.B. durch Verrauschung).

Um diesen Effekt zu limitieren, kann folgende Strategie angewendet werden:

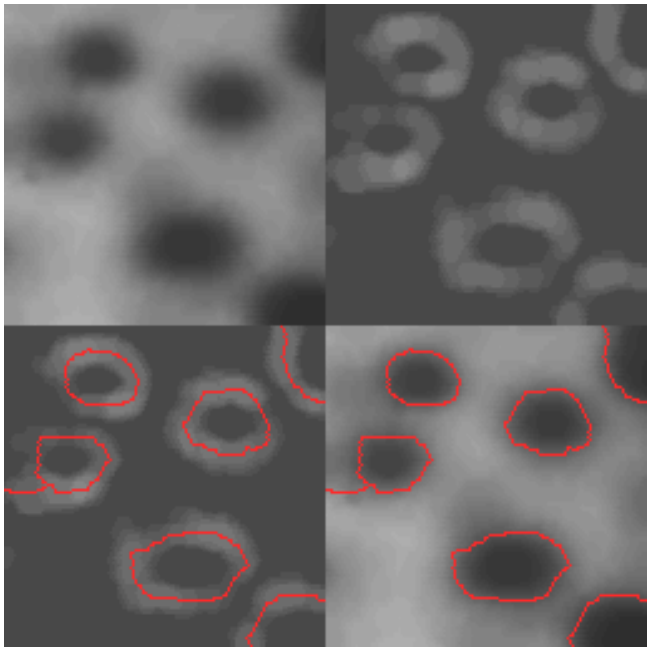


Figure 123: Original, Grandientenbild, Watersheds, original mit Watersheds

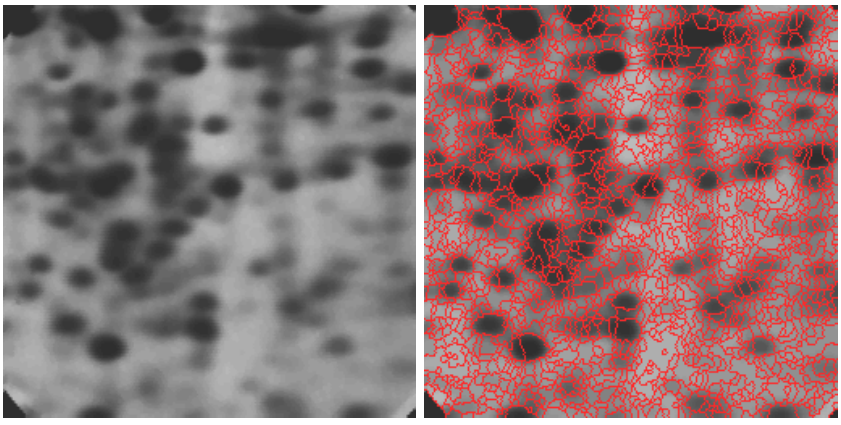


Figure 124: Übersegmentierung

- Glättung (z.B. Gaussfilterung mit grossem σ
- Marker: Es werden als lokale Minima nur sog. "interne Marker" zugelassen; das sind zusammenhängende Regionen mit identischem Grauwert, die von Pixeln mit höherem Wert umgeben sind.

Fig. 125 links zeigt diese inneren Marker. Dann wird der Watershed Algorithmus auf das Bild angewendet. Die resultierenden Watershed Linien werden als "externe Marker" bezeichnet, die das Bild in Regionen partitionieren wobei jede dieser Regionen ein einzelnes Objekt und seinen Hintergrund enthält. Nun kann auf jede einzelne dieser Regionen ein Watershed Verfahren odaer auch z.B. Thresholding angewendet werden, um die gewünschte Segmentierung zu erhalten. Fig. 125 rechts und 126.d zeigt entsprechende Ergebnisse. Zusätzlich kann die Anzahl der inneren Marker beschränkt werden oder iene Mindestgrösse verlangt werden.

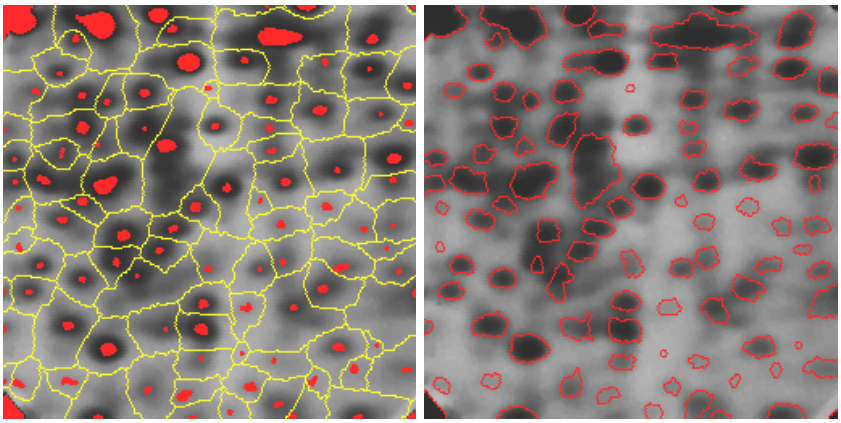


Figure 125: Watersheds mit inneren und äusseren Markern

Dammkonstruktion

Wird beim flooding anders als im skizzierten Algorithmus eine explizite Dammkonstruktion benötigt, wird beim letzten flooding-Schritt $n-1$ vor einem Verschmelzen von zwei Auffangbecken gestoppt (zwei schwarze Regionen in Fig. 127). Die verschmolzene Region nach dem Schritt n bezeichnen wir als q (dargestellt in weiss). Nun wird auf die beiden schwarzen Regionen eine dilation mit einem konstanten 1 3x3 Strukturelement durchgeführt die zwei Bedingungen genügen muss:

1. Das Zentrum des Strukturelements muss in q bleiben.
2. Dilation darf nur durchgeführt werden ohne dass ein Verschmelzen der Regionen geschieht.

Die erste dilation ist im Beispiel unproblematisch, die beiden Regionen werden gleichmässig vergrössert. Im zweiten dilation Schritt erfüllen einige Punkte die erste Bedingung nicht mehr, daher der unterbruchene Umfang. Die Punkte die die erste Bedingung erfüllen und die zweite nicht, sind dann die Dammpunkte.

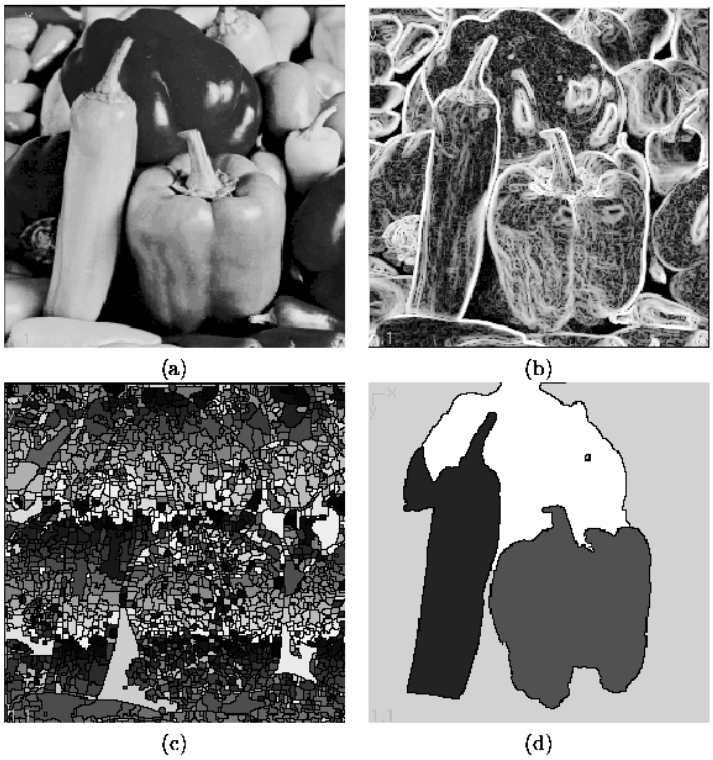


Figure 126: Beispiel: alle Varianten

Diese werden auf den maximalen Helligkeitswert im Bild gesetzt um nicht wieder überflutet zu werden und dann wird das flooding fortgesetzt.

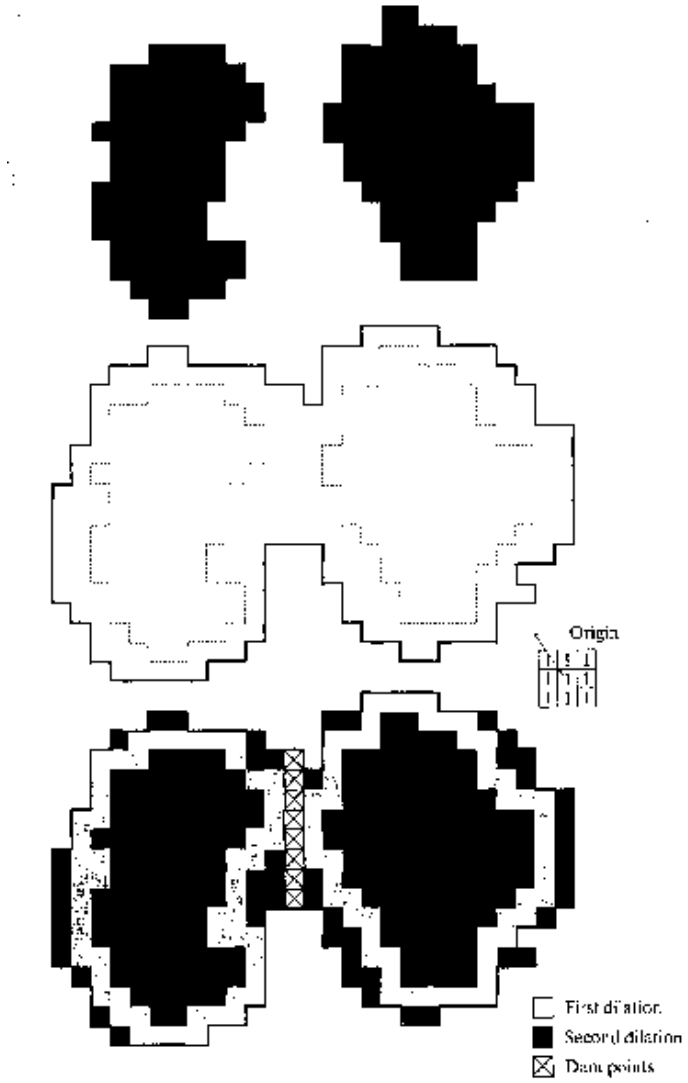


Figure 127: Dammkonstruktion mit Dilation

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.

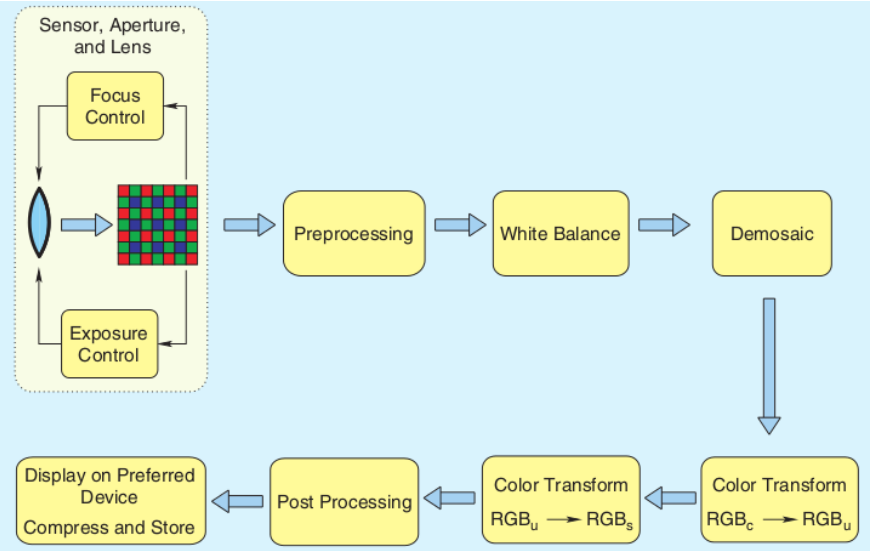


Figure 128: Color Imaging Pipeline: Coarse View.

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.

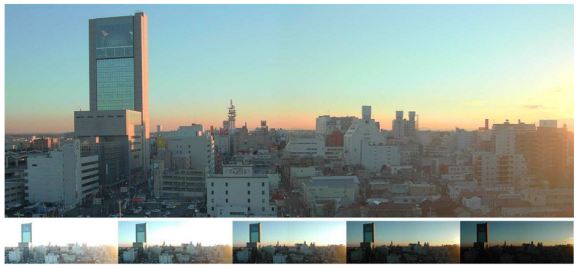


Figure 129: Fusing images with different exposure.

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. 130(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. 130(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.

Fig. 131 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.

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

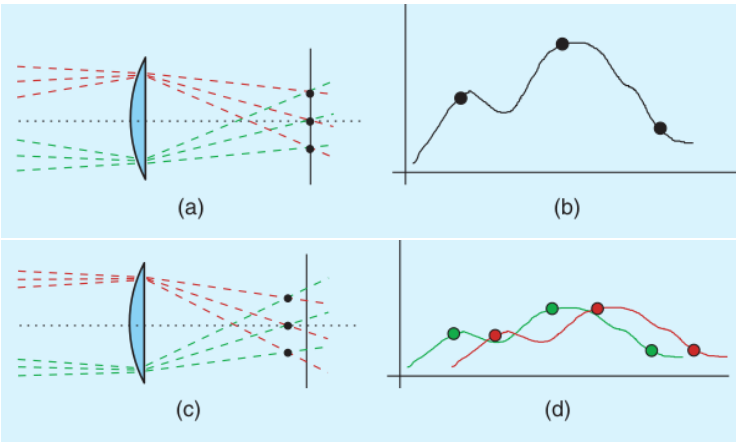


Figure 130: AF phase detection principle.

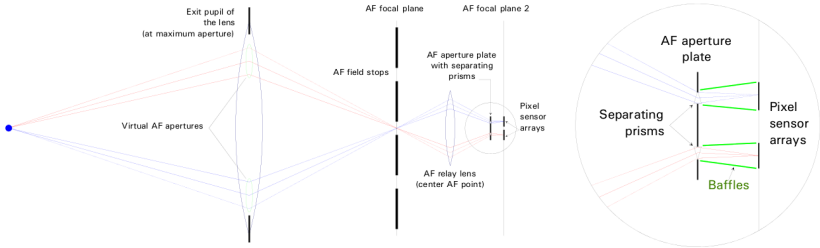


Figure 131: AF phase detection as used in SLR.

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:

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

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:

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

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. 132 for the Laplace function with $n = 1$ and $n = 10$, all examples computed from sequences of 40 hardness testing images with different focus.

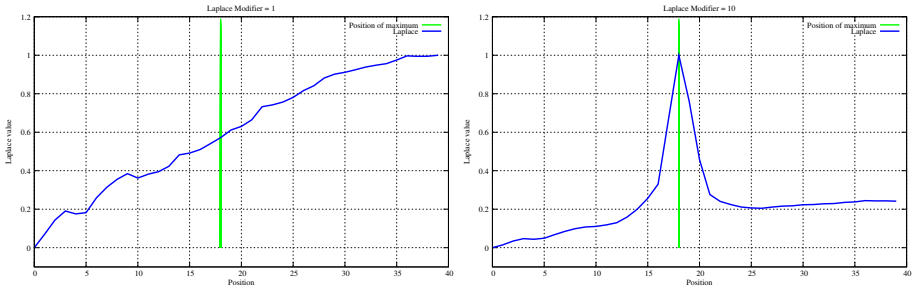


Figure 132: Laplace autofocus function after normalisation applied to a series of 40 images ($n = 1$, $n = 10$).

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. 133 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

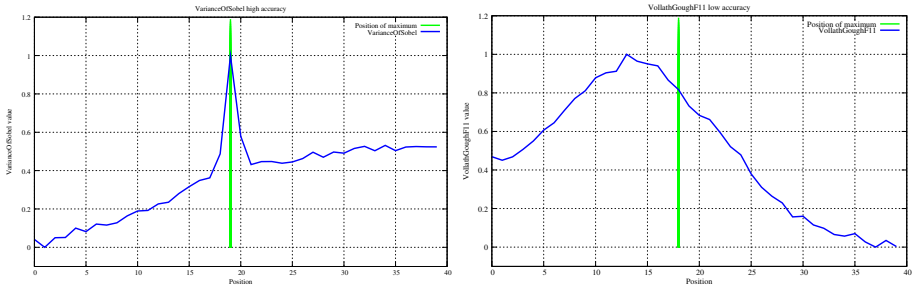


Figure 133: Accuracy of autofocus function: F_{Var_Sobel}, $F_{acc} = 1$, vs. $F_{VollF11}$, $F_{acc} = 0.138$.

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. 134 shows an example with high and poor monotonicity.

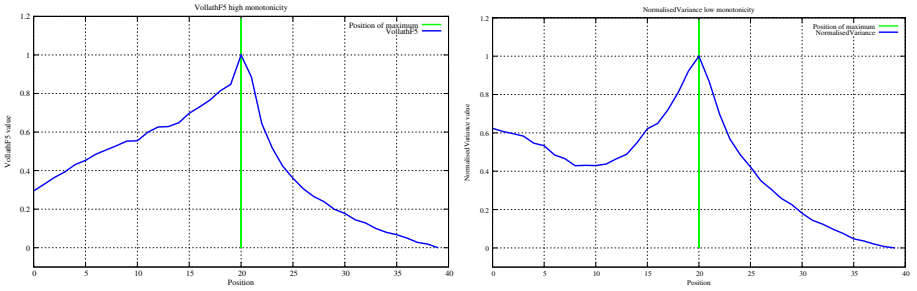


Figure 134: Monotonicity of autofocus function: F_{VollF5}, $F_{mon} = 1$ vs. $F_{Nor\_Variance}$, $F_{mon} = 0.325$.

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. 135 shows an example with high and poor peak sharpness.

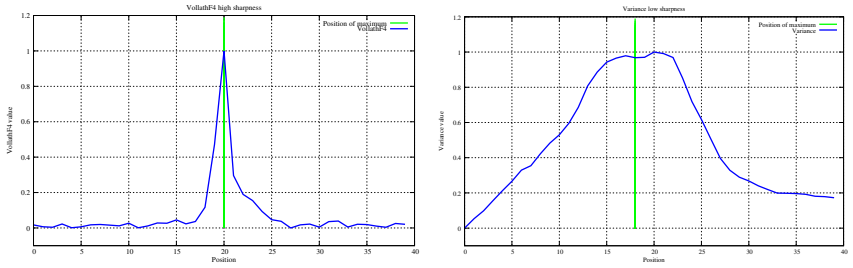


Figure 135: Peak Sharpness of autofocus function: F_{VollF4}, $F_{sharp} = 1$ vs. $F_{Variance}$, $F_{sharp} = 0$.

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. 136, "camera correction", is actually a collection of blocks as detailed in Fig. 137. 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. 137.

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

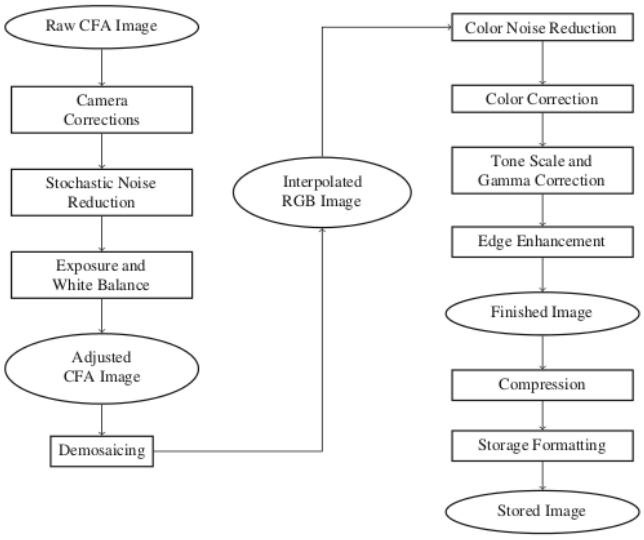


Figure 136: Color Imaging Pipeline: Detailed View.

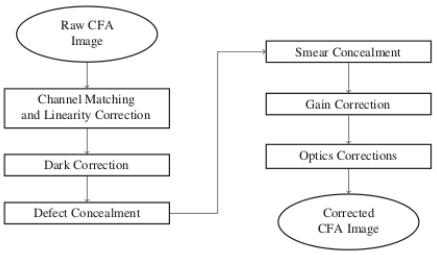


Figure 137: The stages of camera correction.

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. th 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. 138 left), also called bilinear interpolation, its separable implementation requires four samples. Here, first columns are interpolated, followed by an interpolation of the lines.

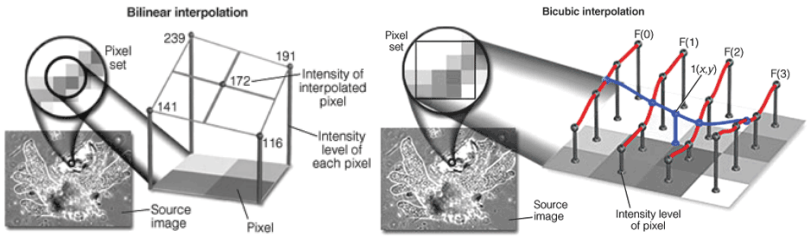


Figure 138: Bilinear and Bicubic interpolation

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 4×4 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. 139. 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): $f_{KEYS}(x) = (a+2)|x|^3 - (a+3)|x|^2 + 1$

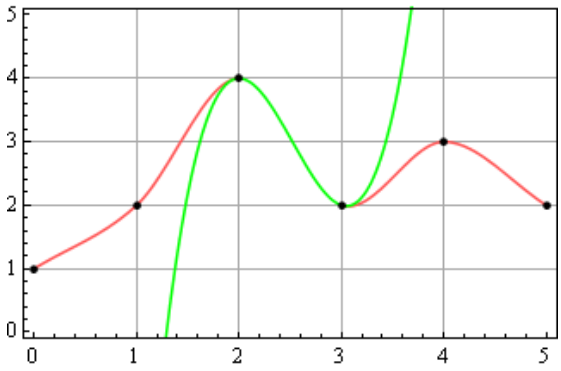


Figure 139: Example for cubic interpolation

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.

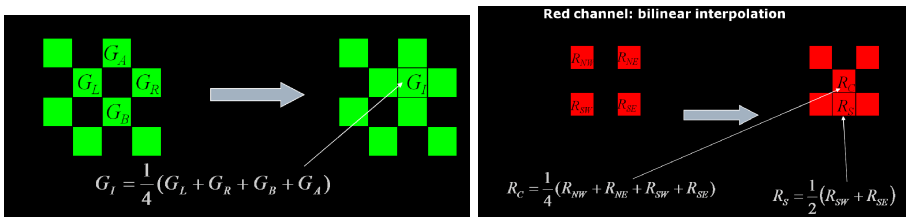


Figure 140: Bilinear colorplane interpolation

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. 140 shows the scheme used for the G and R,B colourplanes, respectively. The examples in Fig. 141 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 141: Examples for color plane interpolation: nearest neighbour vs. bilinear

effect is shown in Fig. 142.

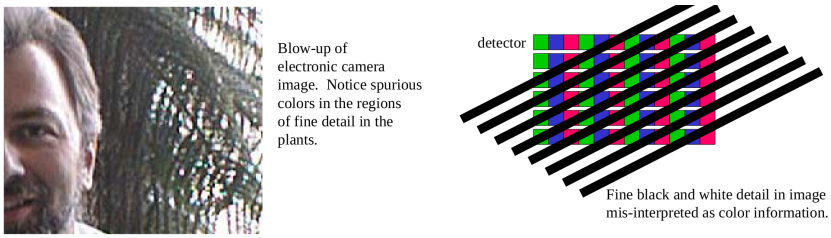


Figure 142: Color Moire artefact

Fig. 143 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.

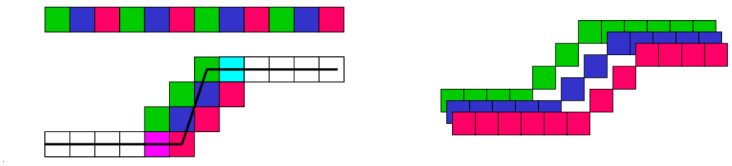


Figure 143: Principle of color sampling errors

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. 144 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. 145 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.

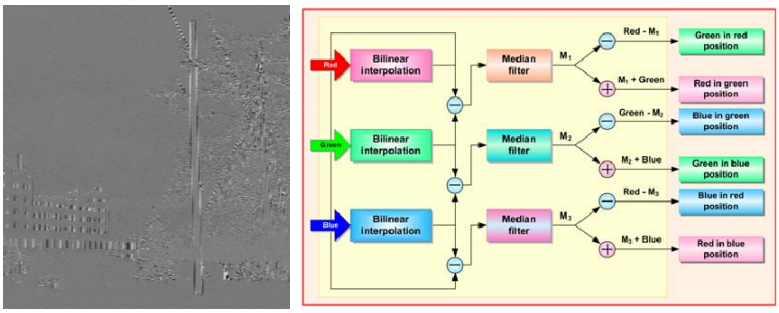


Figure 144: Median filtering approach

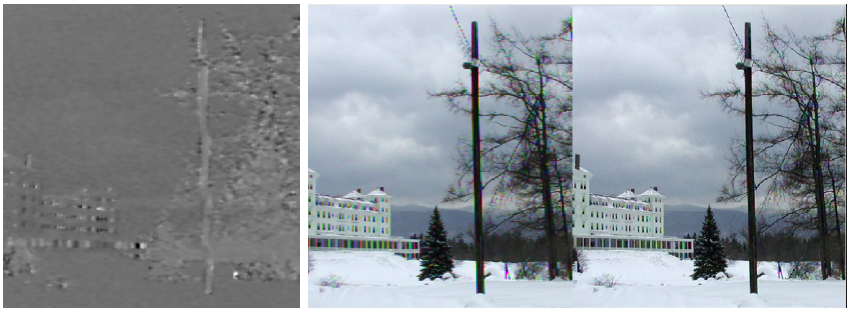


Figure 145: Median filtering result

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 146: Original image and G plane

Fig. 146 is a natural image example which is highly saturated in red and Fig. 147 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 147: 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. 148. 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.

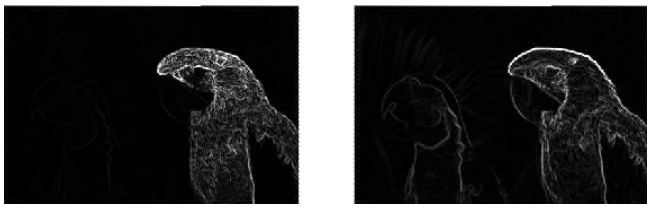


Figure 148: Sobel filter output of R/G and R-G planes

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. 149, 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. 149 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.

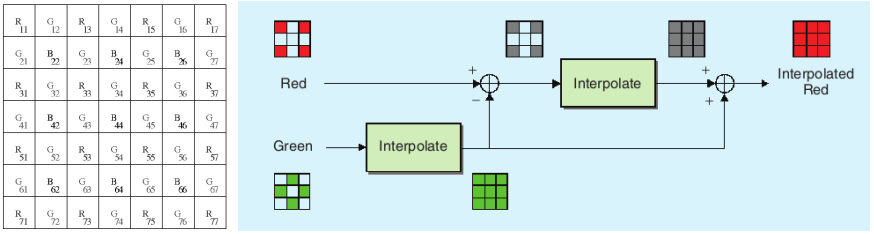


Figure 149: Bayer CFA pattern and constant-difference-based interpolation

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. 150 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.

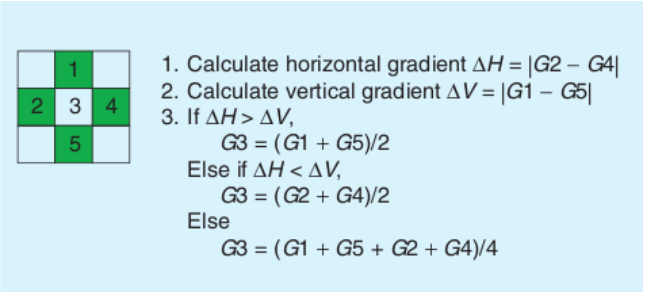


Figure 150: Edge-directed interpolation on a single colour plane.

In Fig. 151, 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. 151 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. 151 is used)

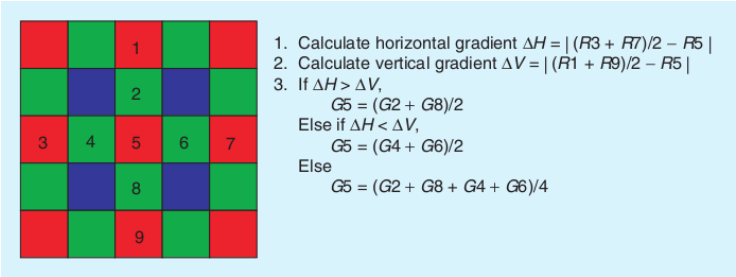


Figure 151: Edge-directed interpolation involving all colour planes.

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

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

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

Here, in fact second order colour gradients are used in the interpolation, in the original scheme, also δV and δH 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. 152 schows the corresponding 8 interpolation schemes (which need to be normalised before application).

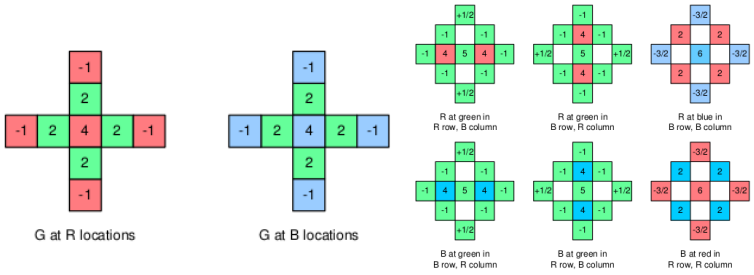


Figure 152: 8 interpolation schemes for High Quality Linear Interpolation

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. 153).

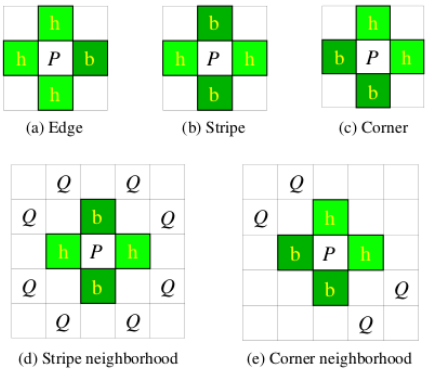


Figure 153: Patterns used to determine the central pixel interpolation

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 (ρ, Φ, ϕ):

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

In the (ρ, Φ, ϕ) space, a filtering operation like median filtering is applied to the angles only. This forces the chrominance components to be similar. Because ρ 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".