Grundlagen Bildverarbeitung

Ao.Univ.-Prof. Dr. Andreas Uhl

WS 2005/2006

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

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.

1.2.2 Low level digital image processing

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.

1.3 Image Properties

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

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

- or higher-dimensional

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

Figure 1: source image

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

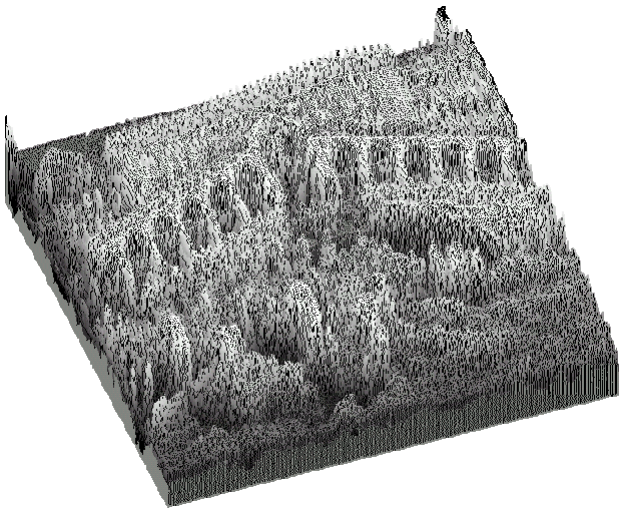


Figure 2: image as 2D-set of brightness values

Figure 3: source image

1.3.1 Color Representation

In the RGB color modell (see figure 4) the luminance value is encoded in each color channel while in the YUV color modell (see figure 5) 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 4: RGB decomposition of figure 3

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

Figure 5: YUV decomposition of figure 3

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

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).

1.3.2 Rasterung und Quantisierung

- Unter Rasterung (siehe figure 6) versteht man die Aufteilung des Bildes (Diskretisierung) in festgelegte Abstände (Anzahl Pixel in beide Richtungen).
- Unter Quantisierung (siehe figure 7) versteht man die Bewertung der Helligkeit 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 6: Rasterung von figure 3

256 colors 64 colors 16 colors

Figure 7: Quantization of figure 3

1.3.3 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 8.

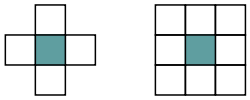


Figure 8: pixel neighborhoods

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

There exist various **contiguity paradoxes** of the square grid as shown in figure 9.

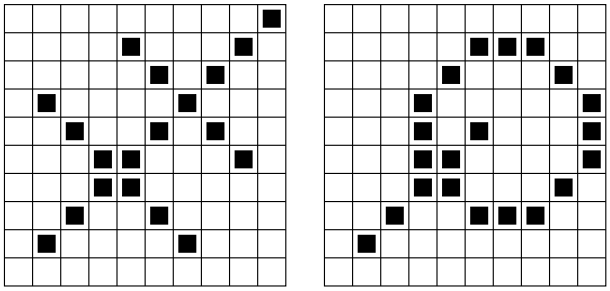


Figure 9: contiguity paradoxes

One possible solution to contiguity paradoxes is to treat objects using 4-neighborhood and background using 8-neighborhood (or vice versa). A hexagonal grid (as depicted in figure 10) solves many problems of the square grids. Any point in the hexagonal raster has the same distance to all its six neighbors.

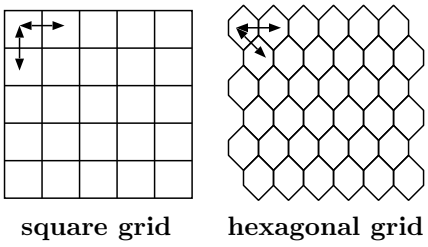


Figure 10: grid types

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 11. The direction of the crack edge is that of increasing brightness, and is a multiple of 90 degrees, while its magnitude is the absolute difference between the brightness of the relevant pair of pixels.

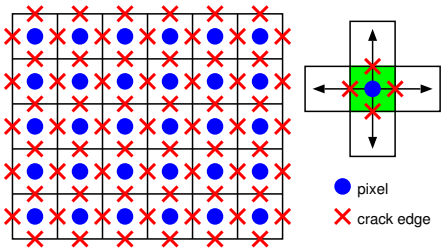


Figure 11: crack edges

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

1.3.4 Histograms

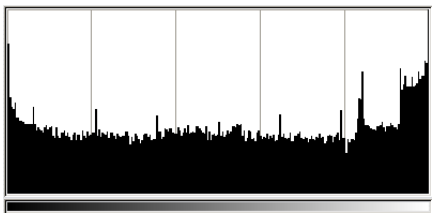


Figure 12: image histogram for figure 3

Brightness histogram provides the frequency of the brightness value z in an image. Figure 12 shows the brightness histogram of the image in figure 3. Histograms lack the positional aspect as depicted in figure 13.

Figure 13: images with same histogram

1.4 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
- ...

1.5 Traditional Data Structures

Some of the tradition data structures used for images are

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

1.5.1 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.

1.5.2 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 14 starting at the red marked pixel is:

00007766555555660000064444444422211111122344445652211

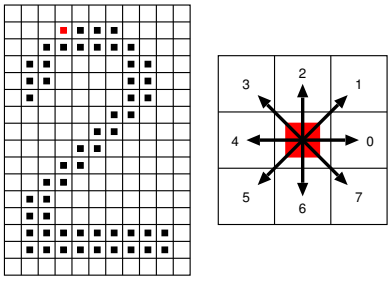


Figure 14: chain data structure

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.

1.5.3 Run length coding

not covered here.

1.5.4 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 15.

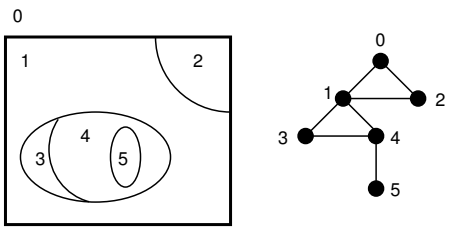
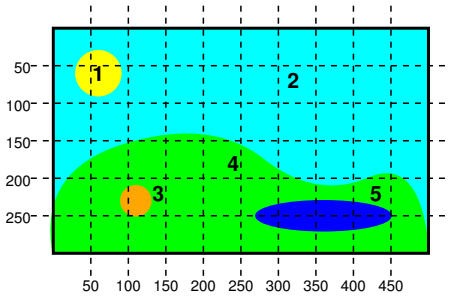


Figure 15: region adjacency graph

1.5.5 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 16.

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



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

Figure 16: relational data structure

1.6 Hierarchical Data Structures

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

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

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

Problems associated with hierarchical image representation are:

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

1.6.1 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 17). Therefore square matrices with dimensions equal to powers of two are required. M_0 corresponds to one pixel only.

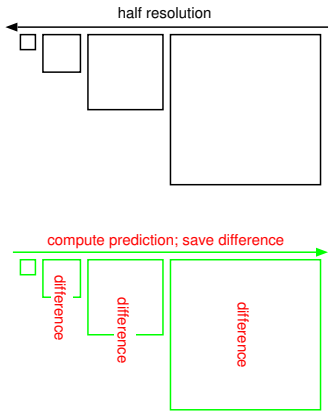


Figure 17: image pyramids

Only the difference between the prediction for M_i (computed from M_{i-1}) and the real M_i is saved for each level i of the pyramid (see figure 17). Methods to reduce the image can be:

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

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

A T-pyramid is a tree, where every node of the T-pyramid has 4 child nodes (as depicted in figure 18).

1.6.2 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 19.

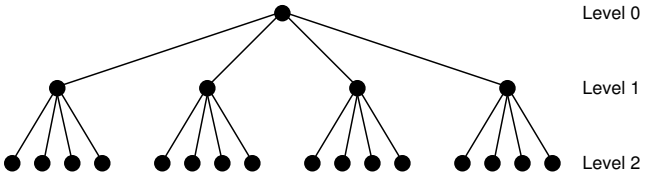


Figure 18: T-pyramid data structure

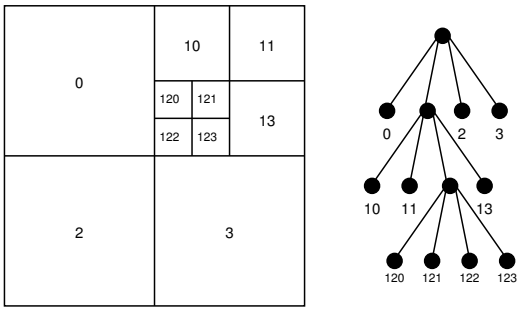


Figure 19: quadtree

1.7 Perception

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

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"

2 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)

2.1 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 20) mit der Form $s = T(v)$ mit $v = f(x, y)$ und $s = g(x, y)$

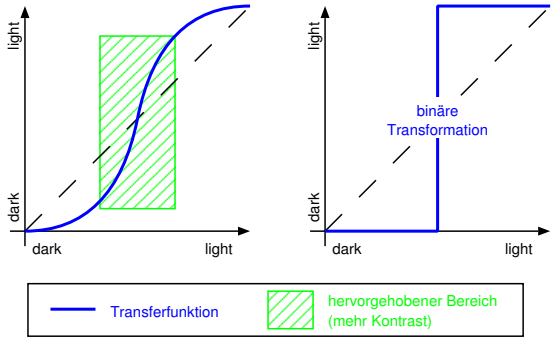


Figure 20: Grauwerttransformation

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

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 21).

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

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

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

	...	$x-1$	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 21: Bild und Maske

2.2 Contrast Manipulierung & Modifizierung

2.2.1 Amplitudenveränderungen

Grauwertbereich verändern (siehe figure 22: Visualisierung von Differenzbildern nach Prediction oder MC; clipping wird gern verwendet wenn wenig Pixel in den Randbereichen des Histogramms liegen - Kontrast wird zusätzlich verbessert).

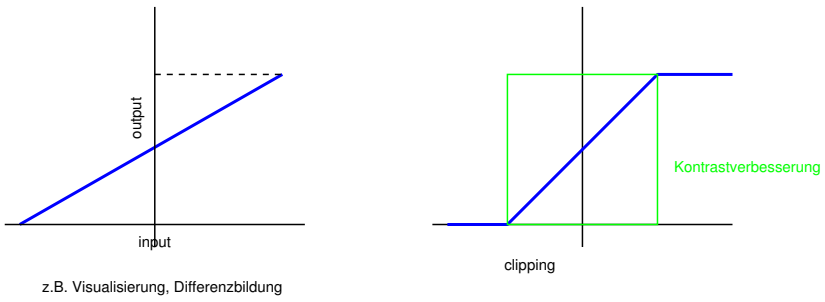


Figure 22: Grauwertbereichs Modifikation

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

2.2.2 Kontrast Modifikationen

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

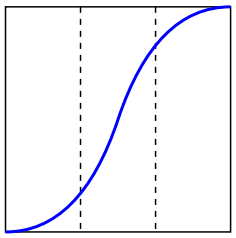


Figure 23: Kontrast Modifikation

Als Beispiel die Kontrast-Modifikation einer Computer Tomographie durch Anwendung des Logarithmus in 24.

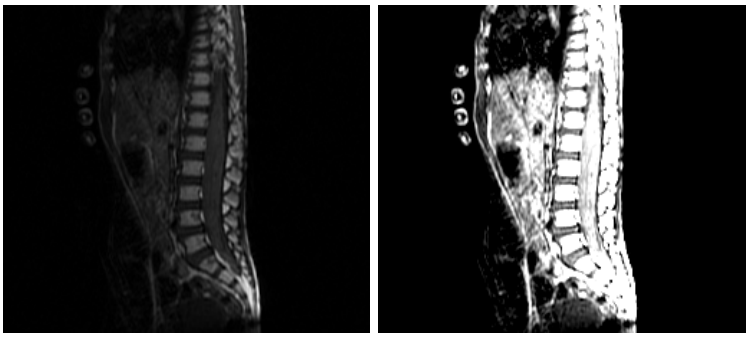


Figure 24: Kontrast Modifikation einer CT

Der Kontrast kann durch einfache Transferfunktionen wie

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

modifiziert werden (siehe auch Figure 23). Typische Kontrastmodifikationen sind in Figure 25 zu sehen.

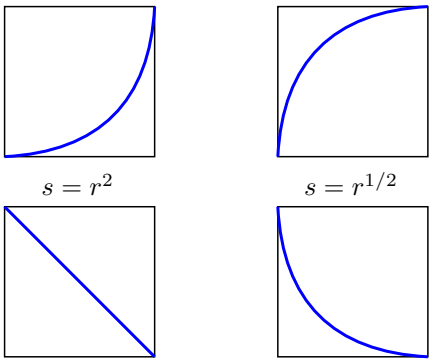


Figure 25: Typische Kontrastmodifikationen

Als Beispiel die Kontrast-Modifikation eines Myelins durch Anwendung des Logarithmus (ähnlich $s = r^{1/2}$) in 26.

2.2.3 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).

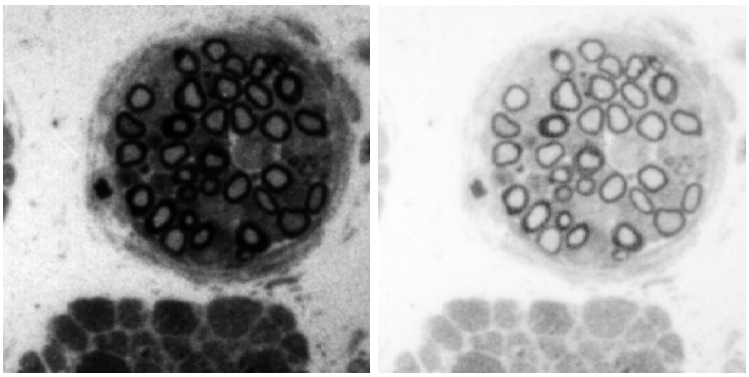


Figure 26: Kontrast Modifikation eines Myelins

Betrachte die Grauwerte als stetige Zufallsvariable. Man kann die originale und transformierte Grauwertverteilung durch ihre Wahrscheinlichkeitsdichten $p_r(r)$ und $p_s(s)$ darstellen. Die Dichten geben Aussagen über das Grundaussehen des Bildes.

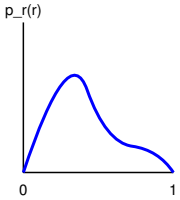


Figure 27: stetiges Histogramm

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:

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

2.2.4 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.

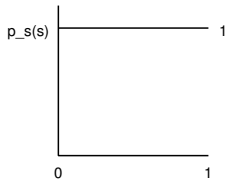


Figure 28: equalisiertes Histogramm

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:

$$\begin{aligned} p_r(r) &= \begin{cases} -2r+2 & 0 \leq r \leq 1 \\ 0 & \text{sonst} \end{cases} \\ s = T(r) &= \int_0^r (-2w+2)dw = -r^2 + 2r \\ r = T^{-1}(s) &= 1 - \sqrt{1-s} \end{aligned}$$

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

Diskretisierung

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

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 29 betrachtet werden. Zu dem Ausgangsbild und dem Histogramm-equalisierten Bild sind auch noch die Histogramme gegeben.

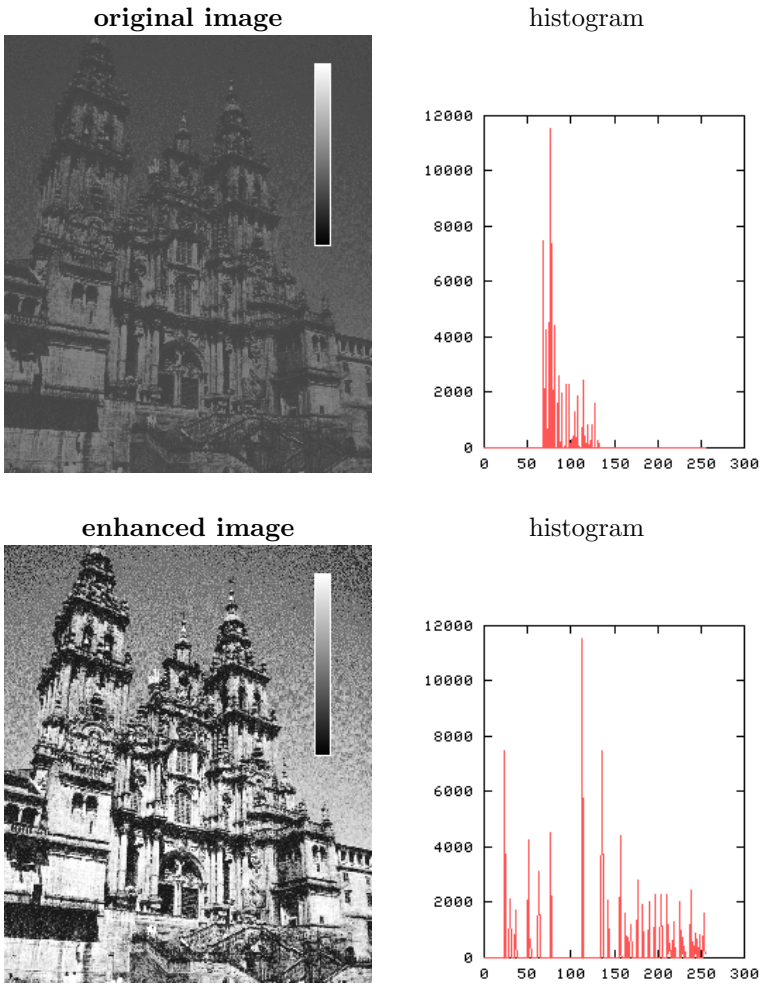


Figure 29: Histogramm Equalisierung

2.2.5 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.

$p_s(s)$ 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.

2.3 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.*

2.3.1 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

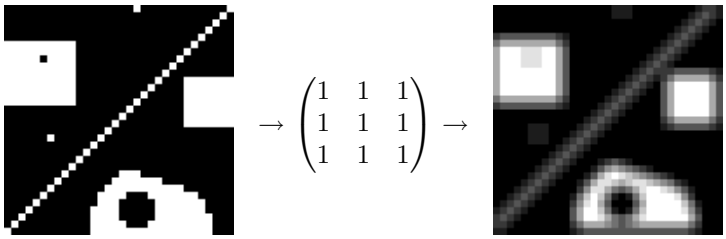


Figure 30: Averaging mit 3×3 Maske

Auch hier können ähnlich figure 21 Masken für die Umgebung verwendet werden:

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.

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

2.3.2 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. 31).

2.4 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".

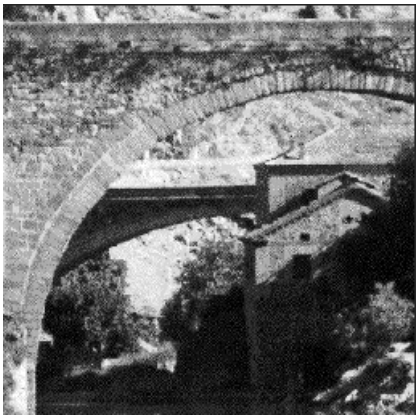
Gradient $G[f(x,y)] = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$

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

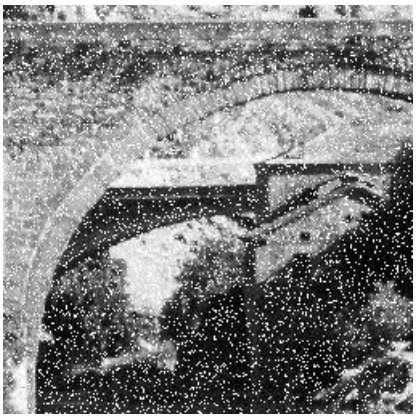
$\text{mag}(G)$ ... Größe (magnitude) von f, gibt maximale Wachstumsrate von $f(x,y)$ an.

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

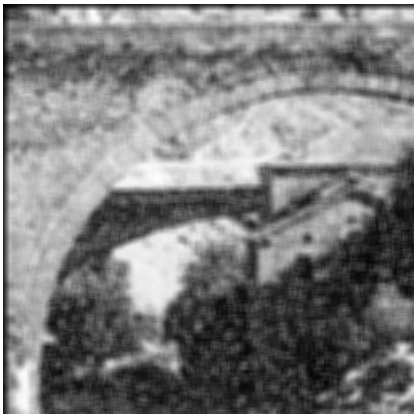
[3] gleichmäßige Dichte



Originalbild



Verrauschtes Bild



5 x 5 Averaging

5 x 5 Median

Figure 31: Denoising

Diskretisierung
Ableitungen werden durch Differenzen approximiert:

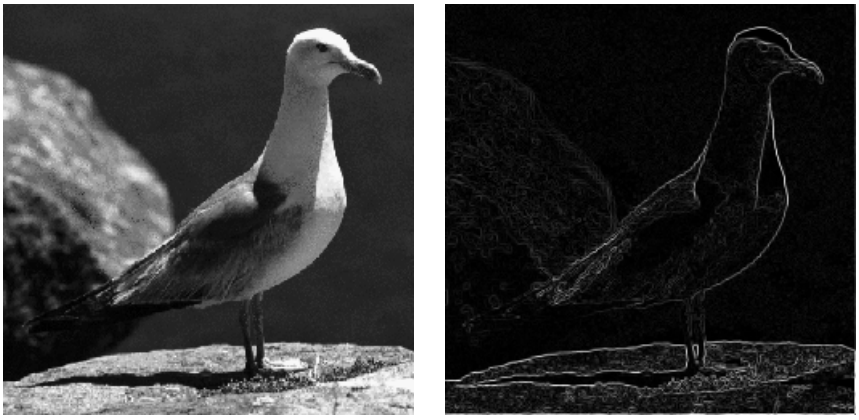
$$|G[f(x,y)]| = \sqrt{[f(x,y) - f(x+1,y)]^2 + [f(x,y) - f(x,y+1)]^2}$$

Alternativ können auch Absolutbeträge verwendet werden (effizientere Implementierung).

Roberts Operator (siehe auch Kapitel 4.1.1) und Fig. 32).

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

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



Originalbild Roberts Gradientenbild

Figure 32: Image Sharpening

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

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

Figure 33: Arten des Gradientenbilds

2.5 Methoden im Transformationsbereich

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

[4] orthogonal und regulär

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

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.

2.5.1 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 34 und 35.

Eigenschaften der 2D Fourier Transformation

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

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

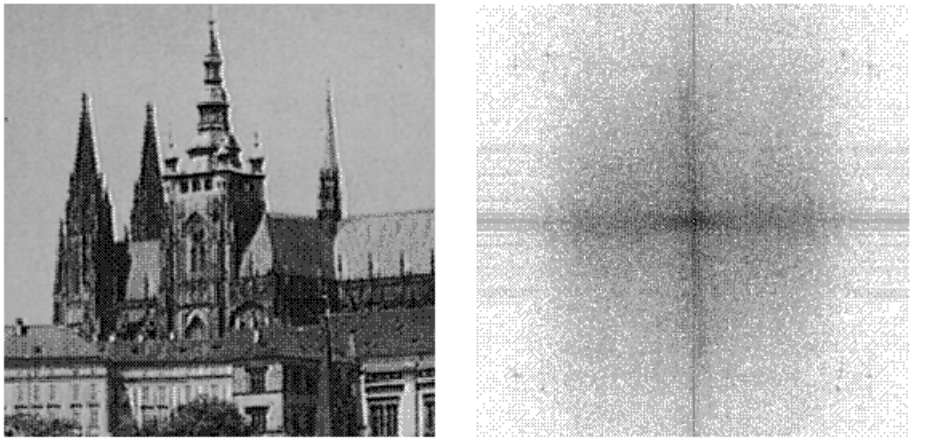


Figure 34: Origial image and its Fourier Spectrum (Magnitude)

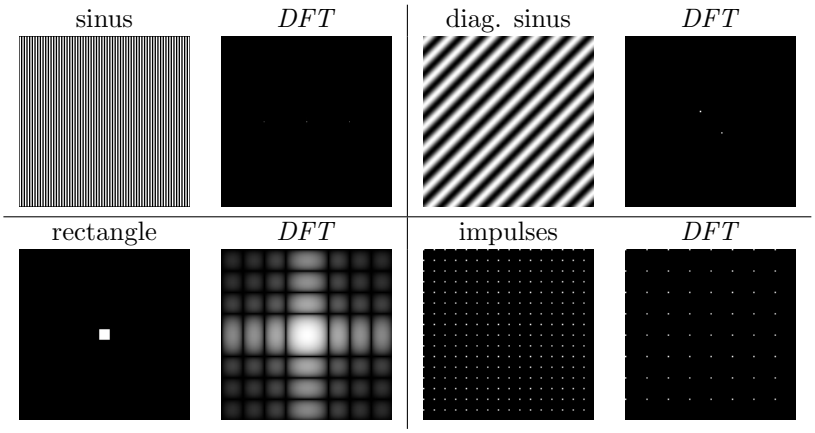


Figure 35: DFT Transformations

- **Separabilität**

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

  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. $e^{-2\pi i(ux+vy)} = e^{-2\pi iux}e^{-2\pi ivy}$.

- **Translation**

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

  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).

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

- **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**

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

- **Skalierung**

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

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 2.5.2)

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.

2.5.2 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 36 sind die im folgenden beschriebenen Filter dargestellt.

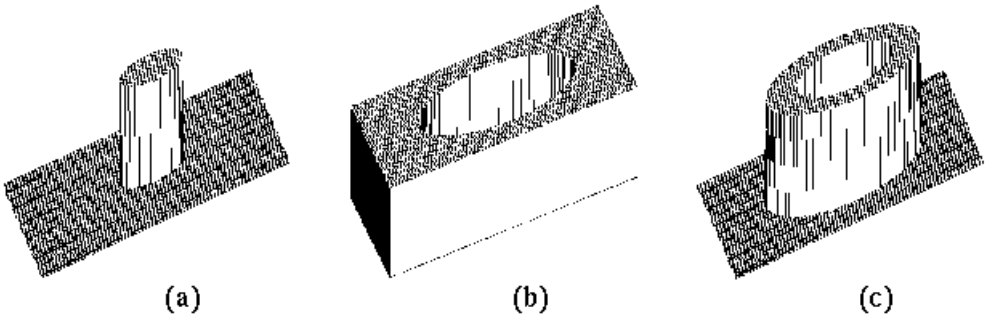


Figure 36: Verschiedene Filter

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. 37). 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.

Butterworth Filter (BLPF)

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

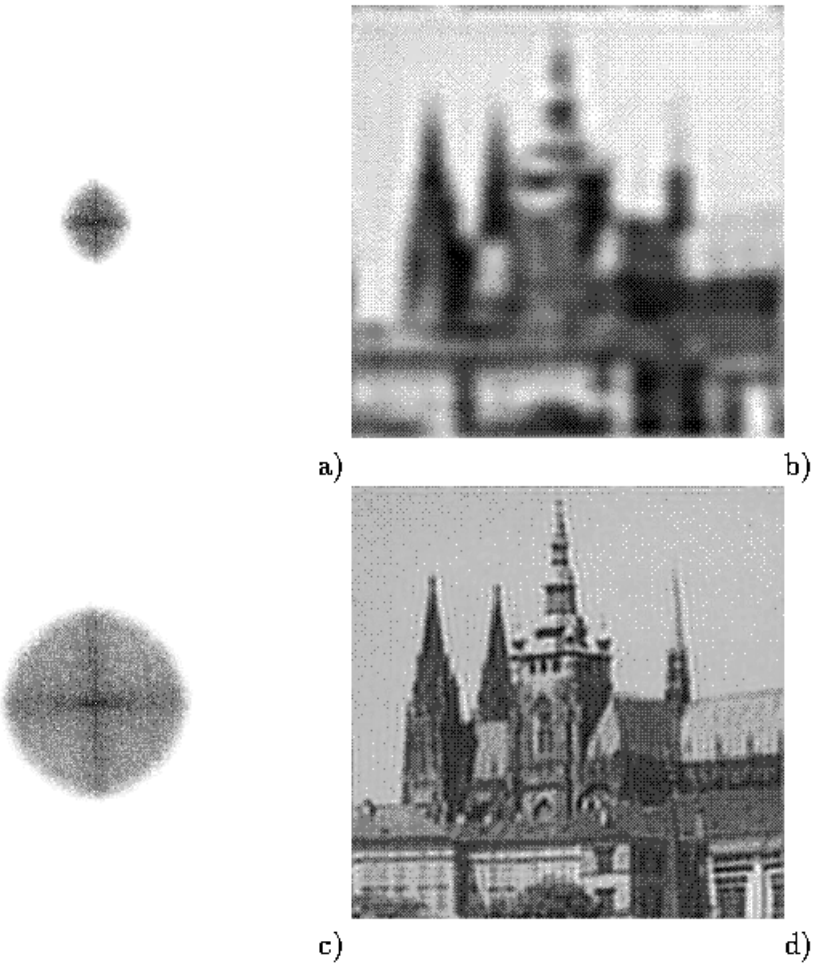


Figure 37: ILPF

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. 38).

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

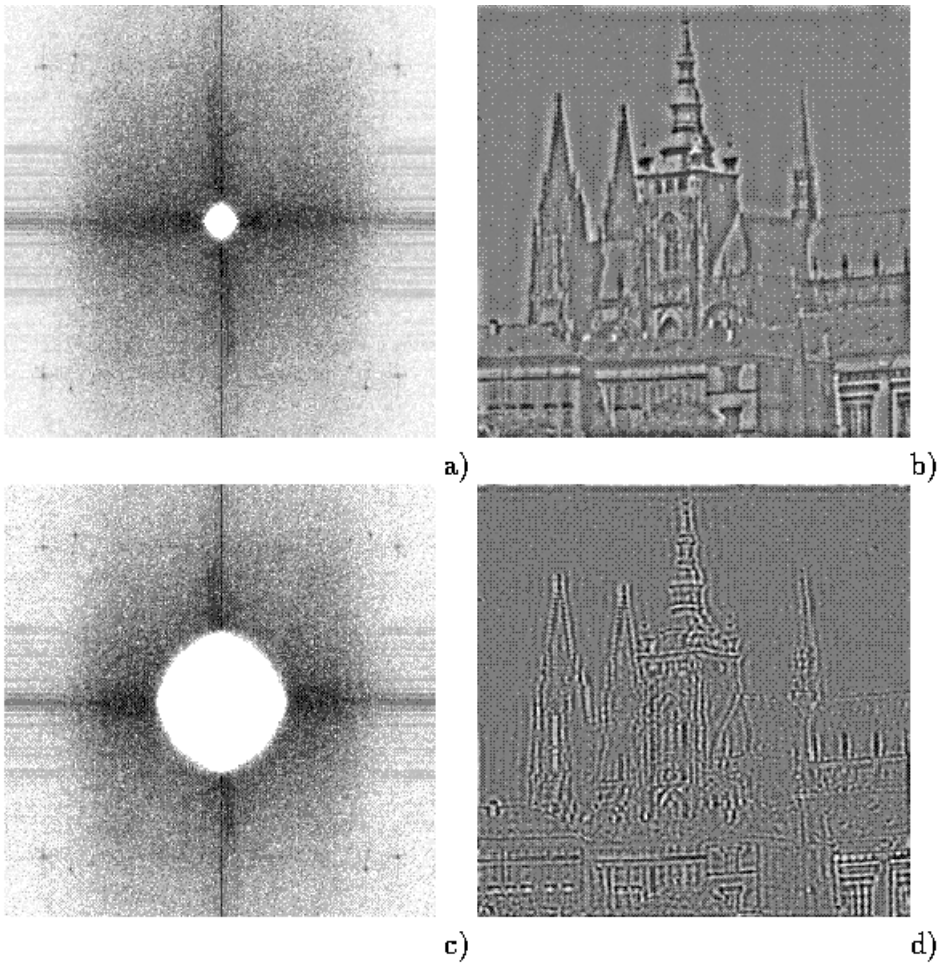


Figure 38: IHPF

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. 39).

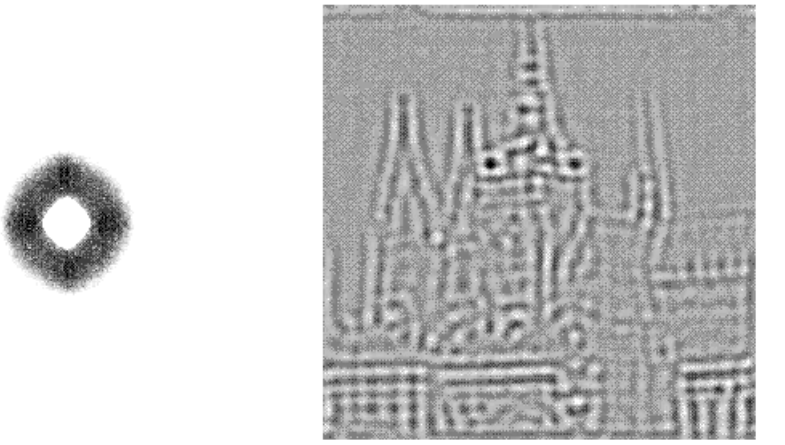


Figure 39: BPF

Speziellere Filterungsmethoden nehmen beispielsweise auf Charakteristika des der Bildstörung zugrundeliegenden Rauschens Rücksicht (siehe z.B. Fig. 40).

2.5.3 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.

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

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

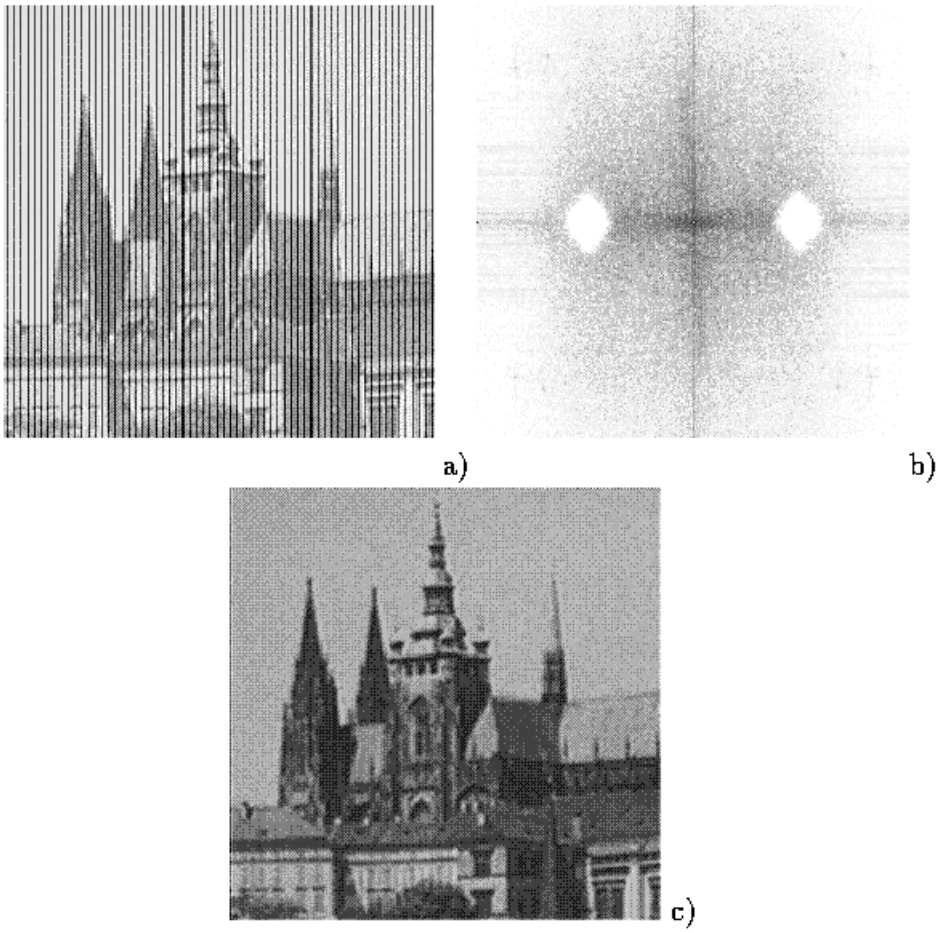


Figure 40: Filterung spezieller Frequenzbereiche

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:

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

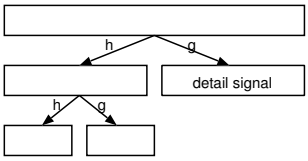


Figure 41: Wavelet Transformation

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 42. Ein Beispiel eines Wavelet transformierten Bildes ist in figure 45 zu sehen.

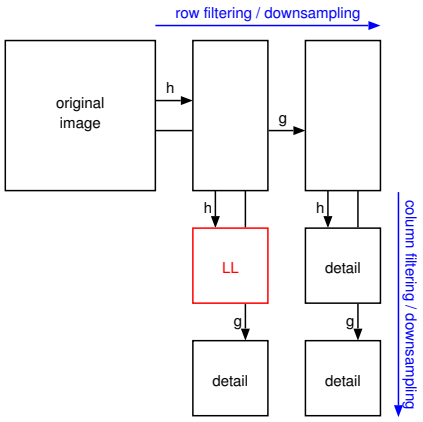


Figure 42: 2D Wavelet Transformation

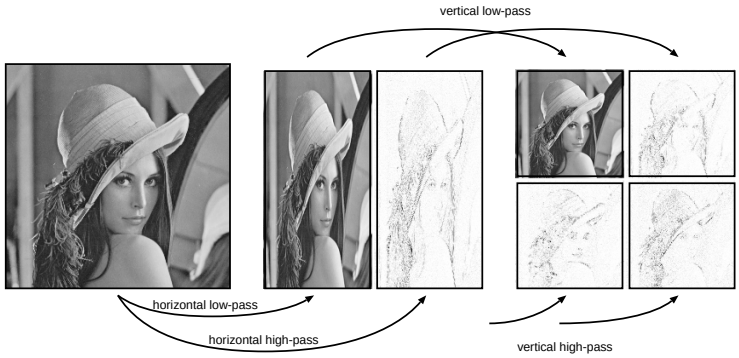


Figure 43: 2D Wavelet Transformation Visualisierung 1. Ebene

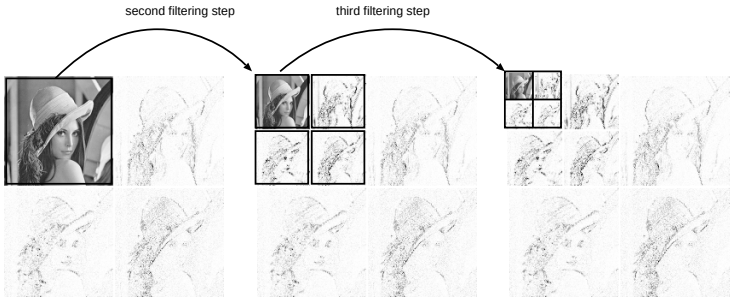


Figure 44: 2D Wavelet Transformation Visualisierung 2.+3. Ebene

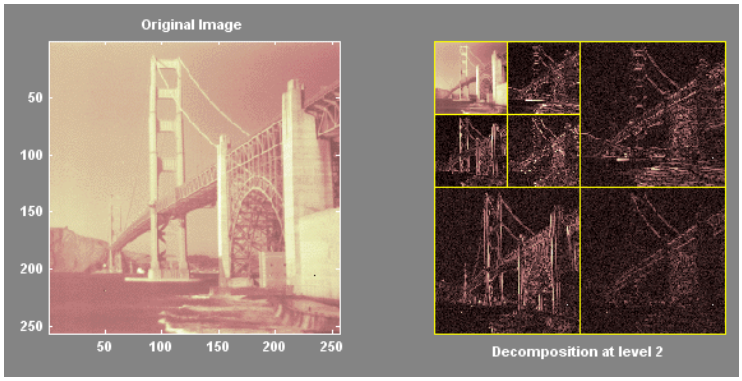


Figure 45: Wavelet Transformation Beispiel

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.

2.5.4 Fourier vs. Wavelet

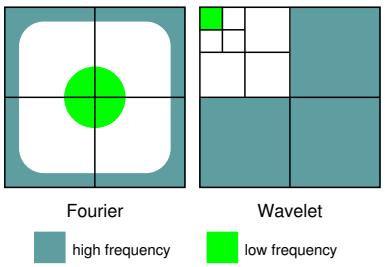


Figure 46: Fourier and Wavelet Transformation

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 46 sind die Anordnung der Frequenzen nach einer der jeweiligen Transformation dargestellt.

2.5.5 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

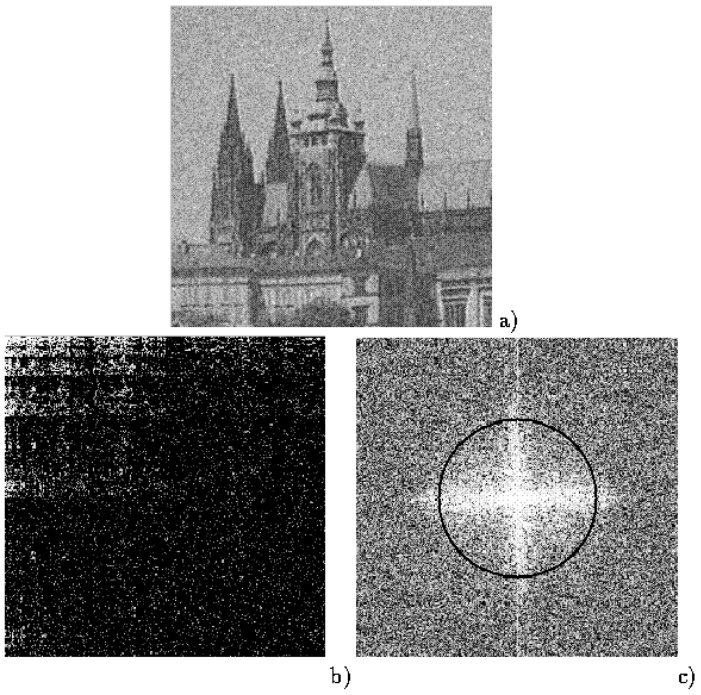


Figure 47: Denoising im Wavelet/Fourier Bereich

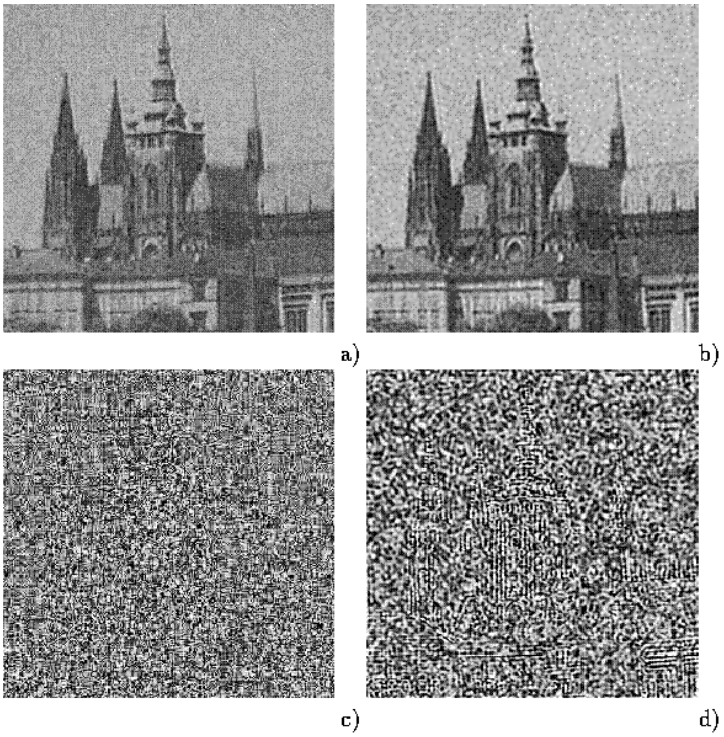


Figure 48: Denoising im Wavelet/Fourier Bereich: Ergebnisse

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. 49). Im Gegensatz zur CWT ist der DWT-Algorithmus einsetzbar. Allerdings wird die Skalierung recht grob gesampled (Oktavschritte).

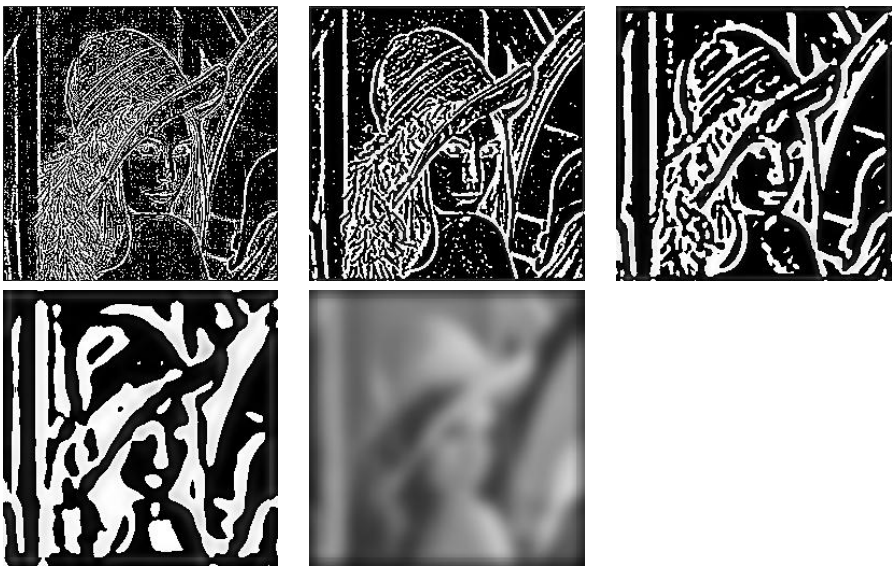


Figure 49: A trous Visualisierung

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.

3 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.

3.1 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:

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

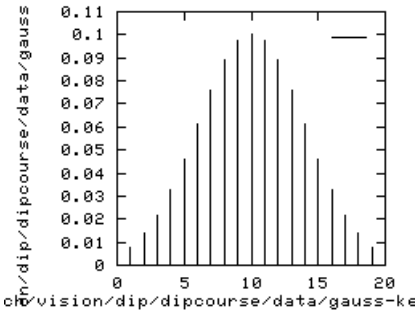
3.2 Bestimmung der Störung

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

3.2.1 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 50: 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.

3.2.2 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:

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

3.2.3 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

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

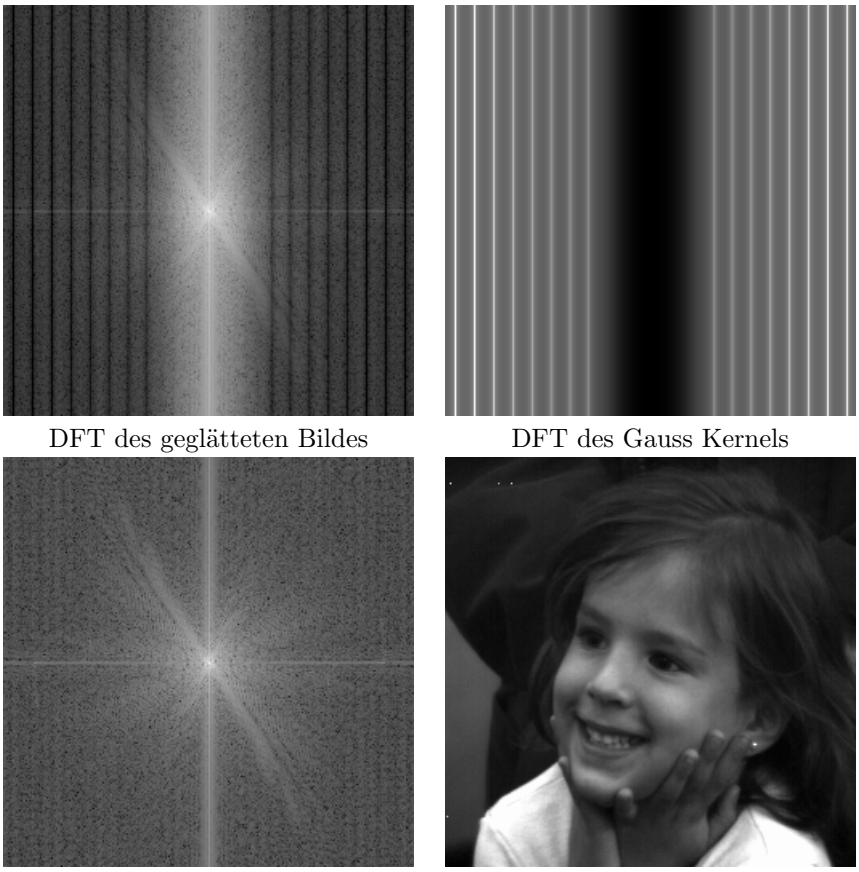
c wird experimentell ermittelt

3.3 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.



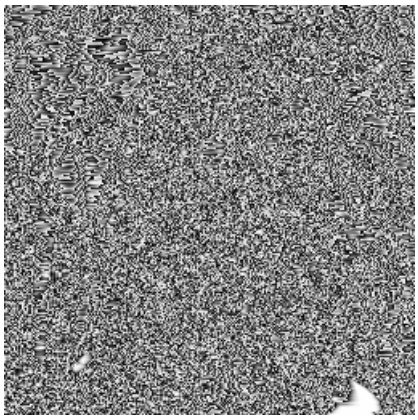
DFT des geglätteten Bildes | DFT des Gauss Kernels

DFT nach inverser Filterung | Restauriertes Bild

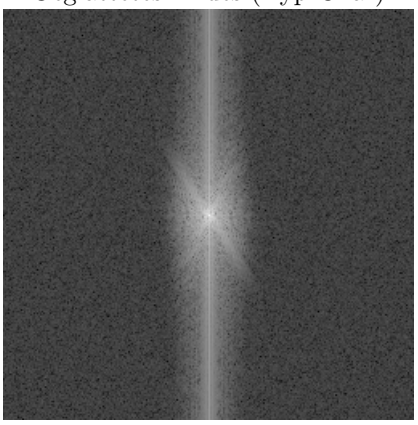
Figure 51: Beispiel Inverse Filterung

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. 52 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.

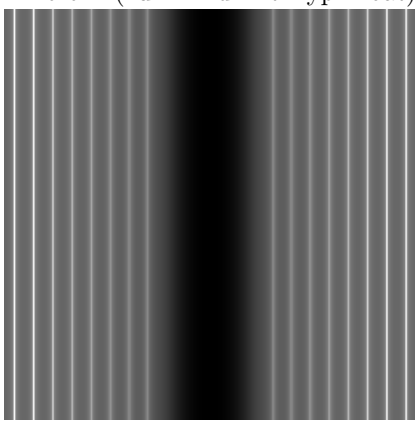
Geglättetes Bildes (Typ Char)



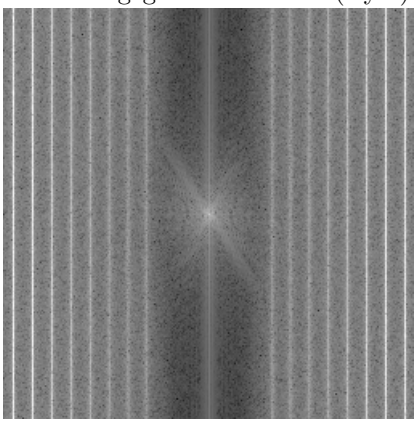
Differenz (zum Bild mit Typ Float)



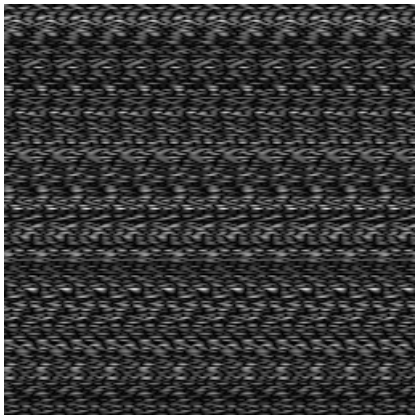
DFT des geglätteten Bildes (Byte)



DFT des Gauss Kernels



DFT nach inverser Filterung



Rekonstruiertes Bild

Figure 52: 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 53: Beispiel Pseudo Inverse Filterung

3.4 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 54 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 54: 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.

4 Kantenerkennung

Die Begriffe Kante (*edge*) und *crack edge* wurden bereits in Kapitel 1.3.3 besprochen.

Zur Erinnerung: Kanten sind Pixel, wo die Helligkeitsfunktion die Größe verändert. Crack edges sind ein Kantenkonstrukt zwischen Pixeln (siehe figure 11).

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

4.1 Methoden mit 1. Ableitung

4.1.1 Roberts Operator

Der Roberts Operator (siehe figure 57) 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.

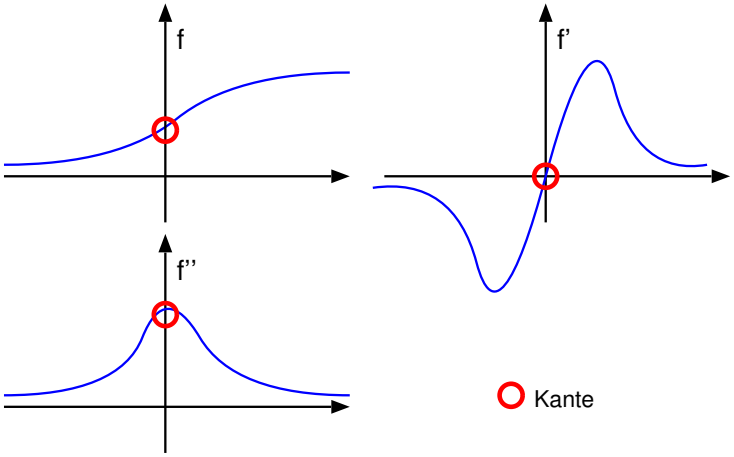


Figure 55: Visuell: 1. Ableitung vs. 2. Ableitung (wer findet die zwei Fehler in der Graphik ?)

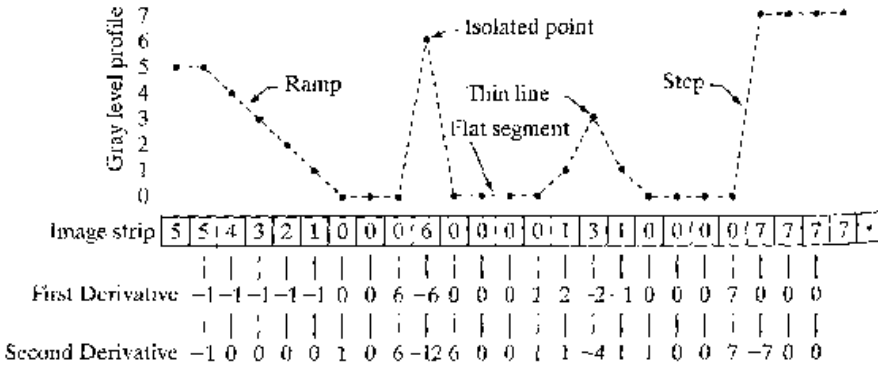


Figure 56: Numerik: 1. vs. 2. Ableitung

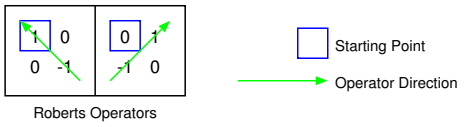


Figure 57: Roberts Operator

4.1.2 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 58)

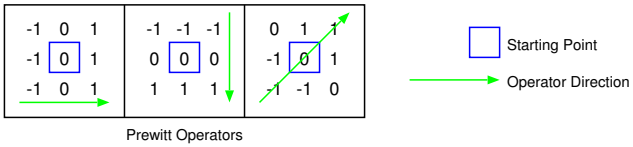


Figure 58: Prewitt Operator

Sobel Operator (siehe figure 59 und figure 60)

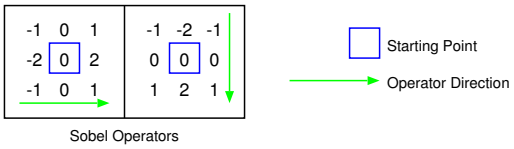


Figure 59: Sobel Operator

Figure 60: 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 64 ist die Anwendung von verschiedenen Kantenerkennungs-Operatoren dargestellt.

Nachteile

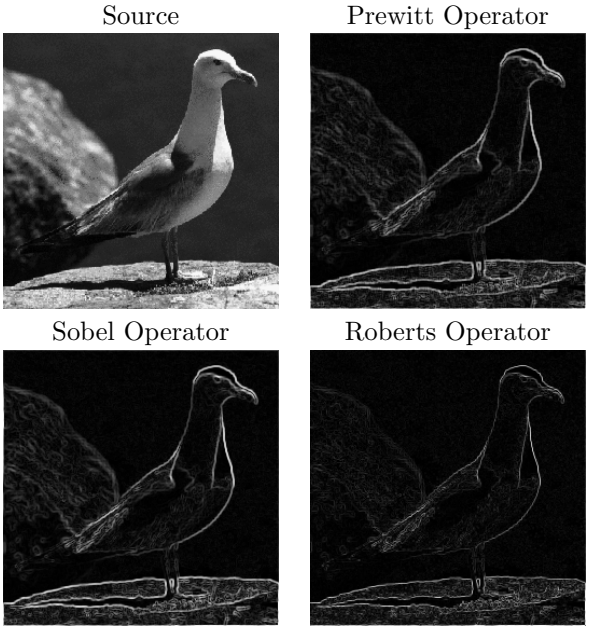


Figure 61: Edge Detection Examples

- 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 56).

4.2 Methoden mit der 2. Ableitung

4.2.1 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

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

4.2.2 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 62) hat.

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

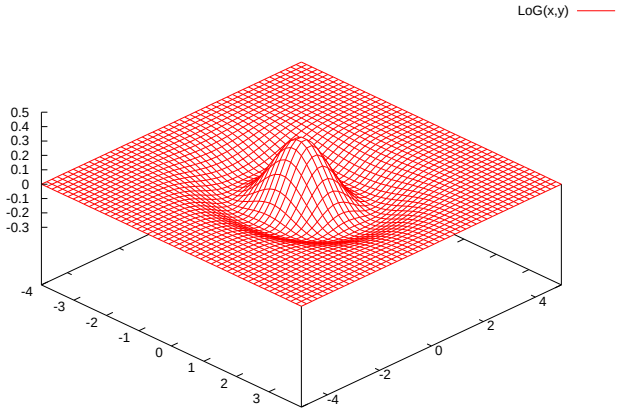


Figure 62: Mexican Hat

Fig. 63 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).

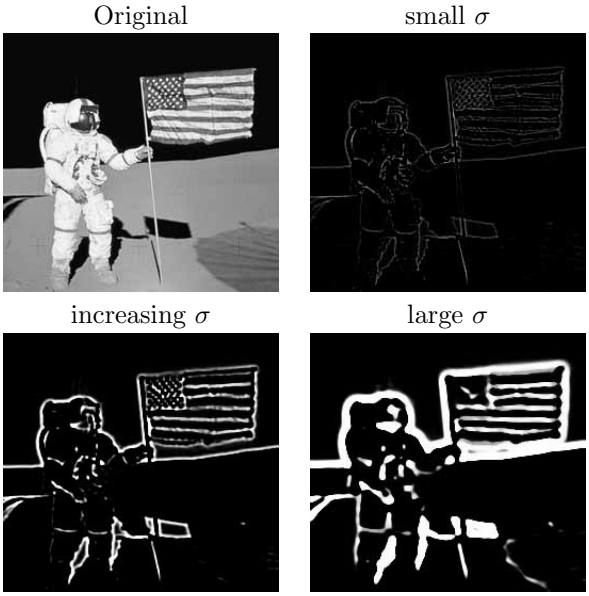


Figure 63: LoG Example

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*)

4.3 Canny Edge Detector

Der Canny Edge Detector (für ein Beispiel siehe figure 65) 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*

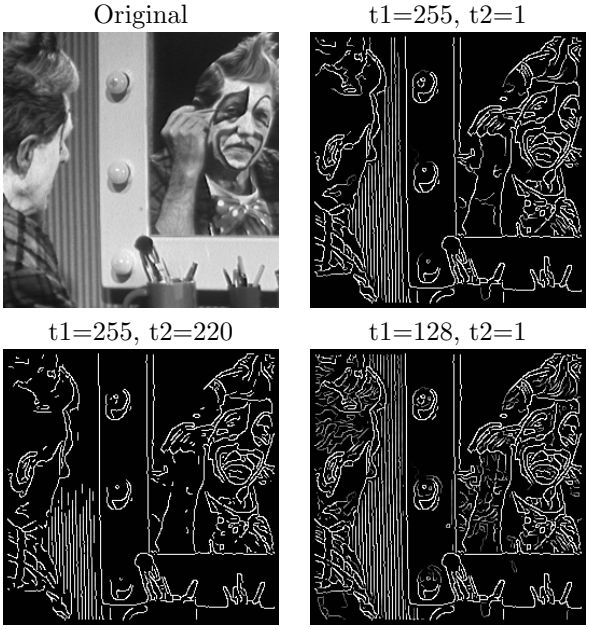


Figure 64: Edge Detection Examples: thresholding

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 4.2.2) 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

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

Figure 65: Canny Edge Detector: verschiedene σ

4.4 Line Finding Alogrithmen

4.4.1 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

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

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

4.4.2 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

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

1. alle Geradenpixel bestimmen (durch Kantenerkennung)

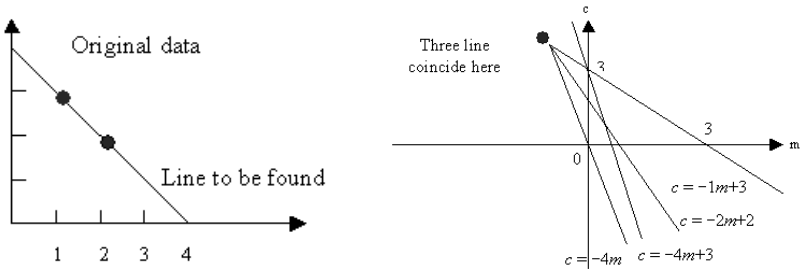


Figure 66: Grundprinzip der Houghtransformation

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.

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 drei-

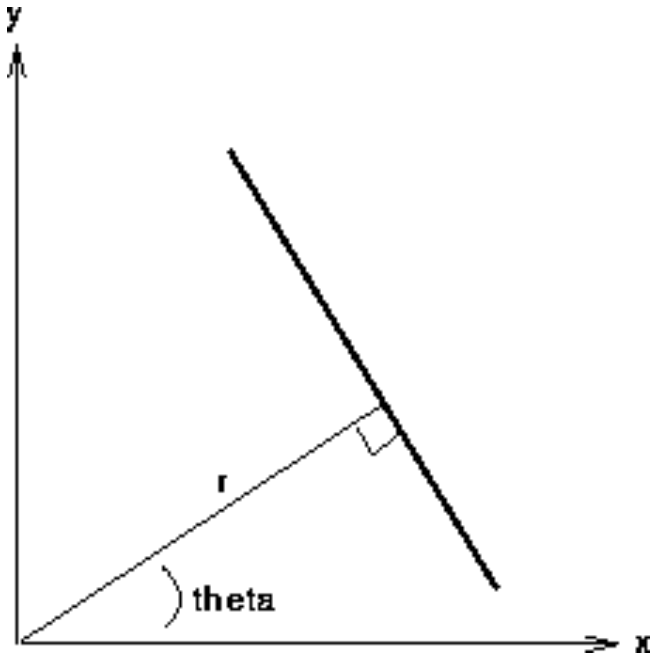


Figure 67: Alternative Geradendarstellung

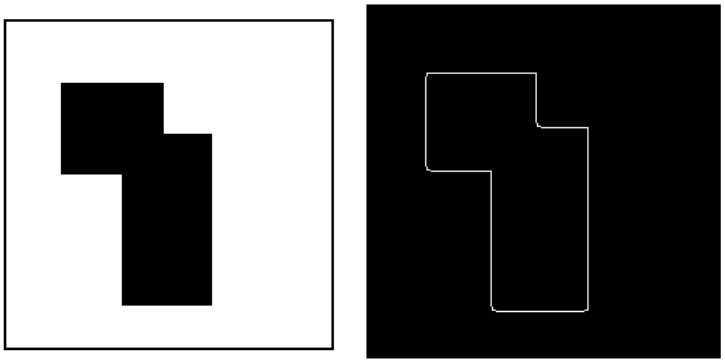


Figure 68: Beispiel der Houghtransformation

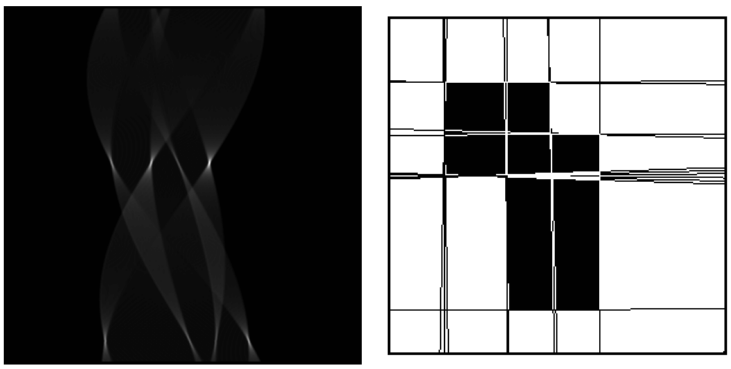


Figure 69: Beispiel der Houghtransformation

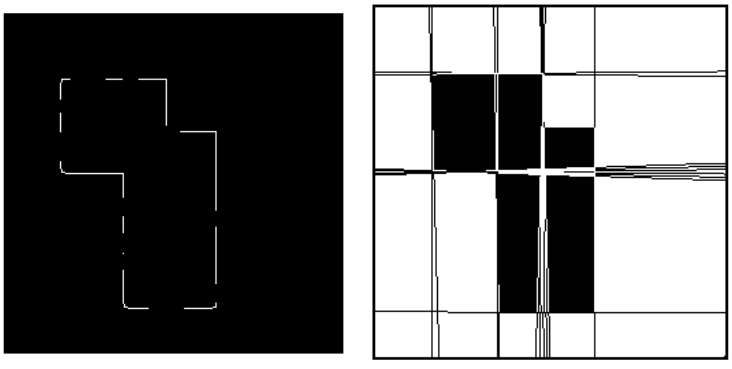


Figure 70: Beispiel der Houghtransformation

dimensionaler Akkumulatorarray benötigt. Für kompliziertere Kurven werden hochdimensionale Akkumulatorarrays verwendet.

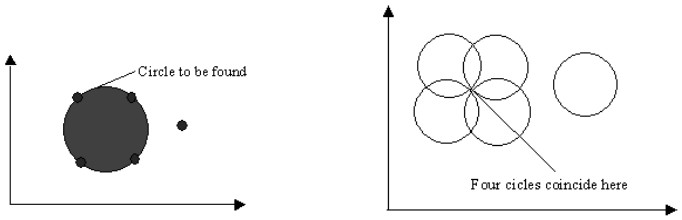


Figure 71: Zirkuläre Houghtransformation

5 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).

5.1 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.

5.2 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

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

5.3 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.

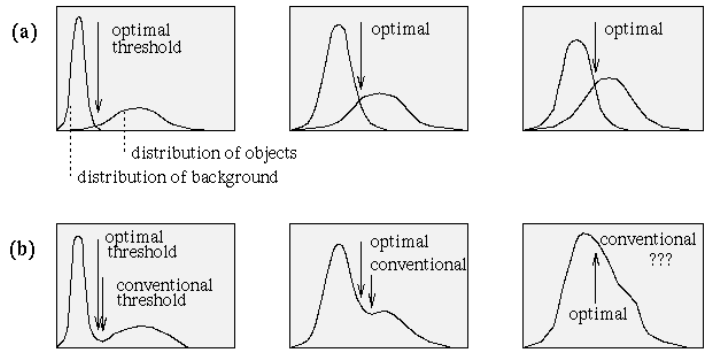


Figure 72: Konzept von optimalen Thresholding

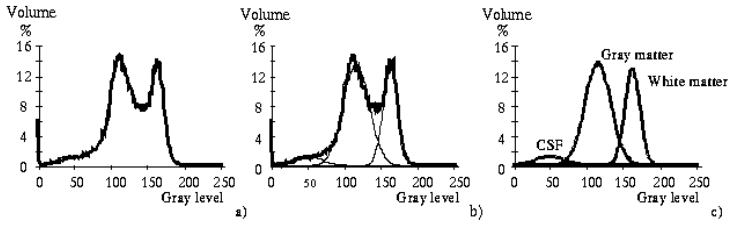


Figure 73: Beispiel für optimales Thresholding: Histogramm

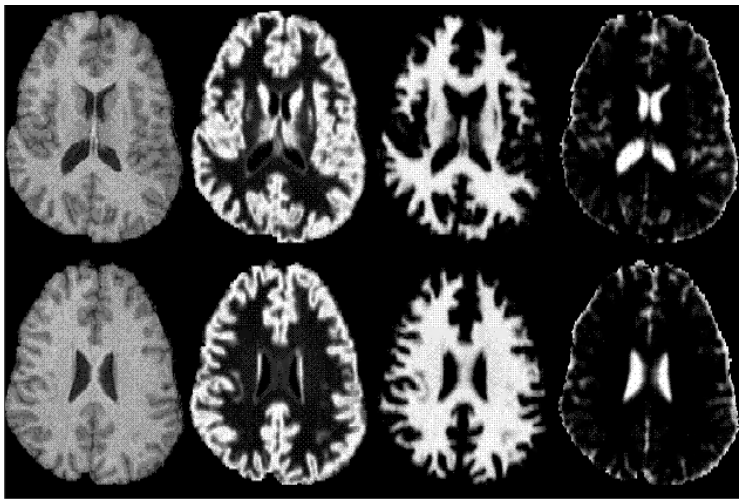


Figure 74: Beispiel für optimales Thresholding: Ergebnis

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.

Thresholding in hierarchischen Datenstrukturen verwendet Pyramidenstruktur (siehe auch Kapitel 1.6). 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.

5.4 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.

5.4.1 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.

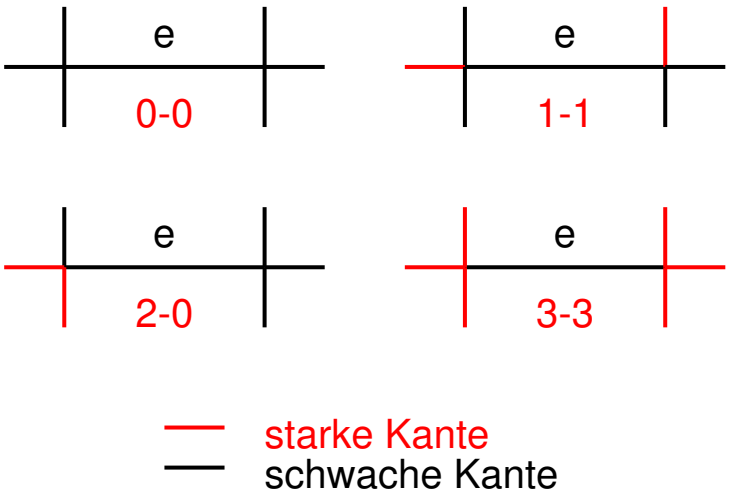
5.4.2 Edge Relaxation

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 1.3.3) 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 75). 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.

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)



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 75: Kanteneigenschaften

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

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

5.4.3 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 $s(x_j) \geq T$.

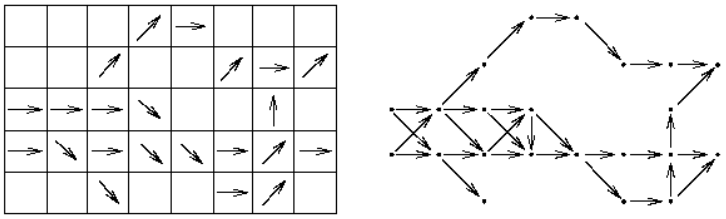


Figure 76: Graphen Repräsentierung eines Kantenbildes

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.

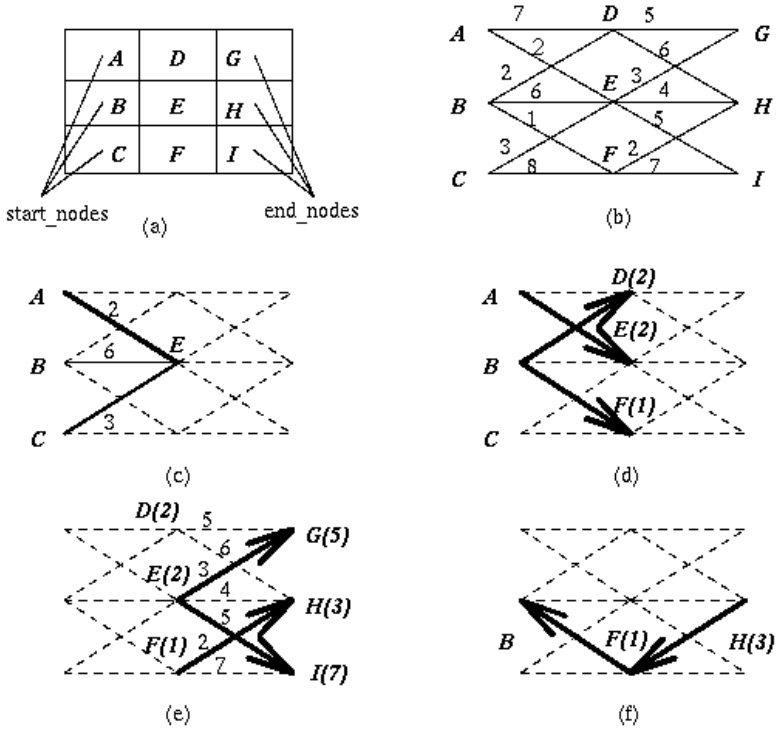


Figure 77: Beispiel Dynamic Programming: (a) Kantenbild (b) Graph mit Kosten (c) mögliche Pfade nach E, A-E ist optimal (d) optimale Pfade nach D,E,F (e) optimale Pfade nach G,H,I (f) Backtracking von H bestimmt Pfad mit geringsten Kosten

5.5 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.

5.5.1 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.

5.5.2 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 78 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.

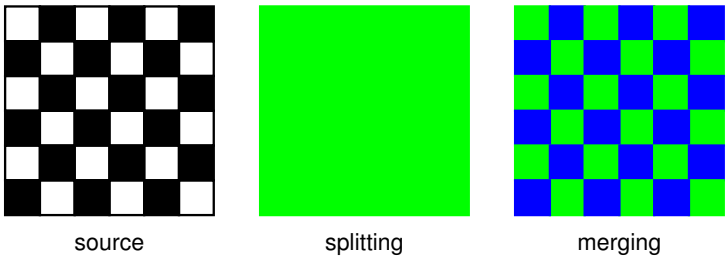


Figure 78: splitting/merging

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.

5.5.3 Template Matching

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

5.6 Watershed Segmentierung

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

6 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 79) 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.

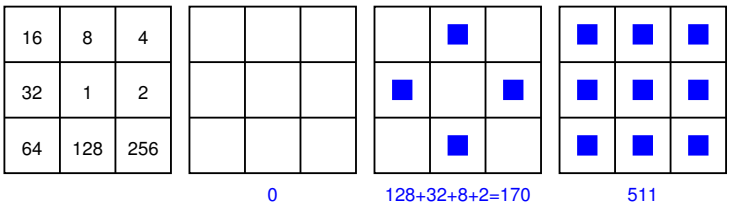


Figure 79: Binary neighborhood encoding

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.

6.1 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 80). 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.

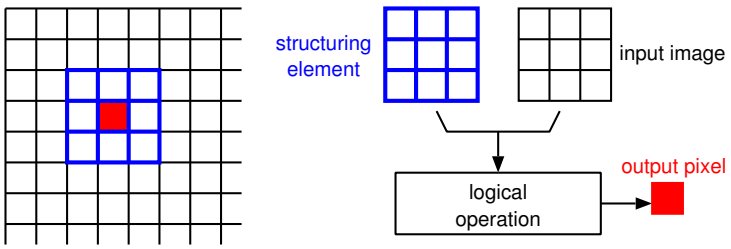


Figure 80: Morphological image processing

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.

6.1.1 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.

6.1.2 Erosion and Dilation

The basic morphological operations are erosion and dilation (see figure 81). By definition, a boundary point is a pixel that is located inside an object, but that has at least one neighbor outside the object.

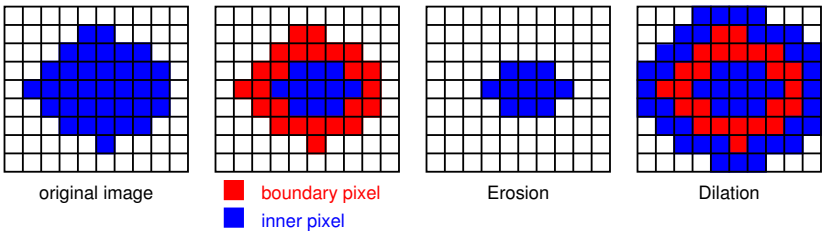


Figure 81: Erosion and dilation

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.

6.1.3 Opening and Closing

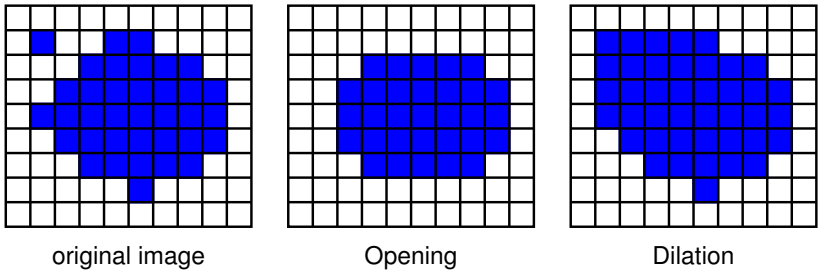


Figure 82: Opening and closing (auch in der Figure !)

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.

6.2 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.

6.3 Thinning

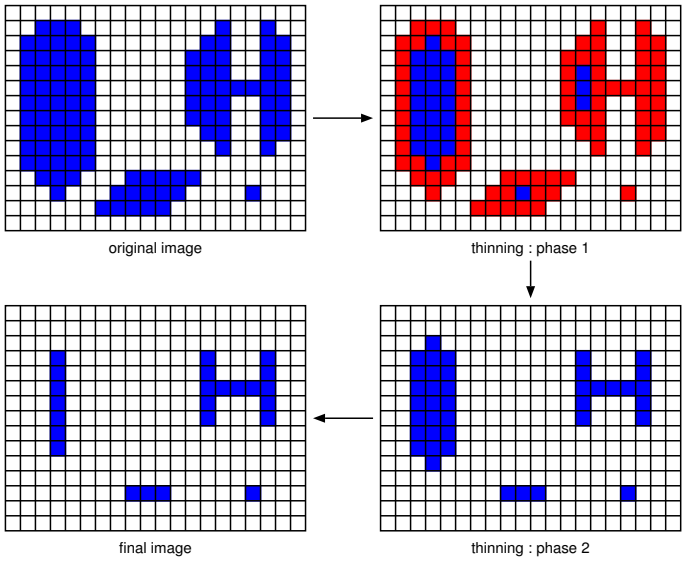


Figure 83: Thinning

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 83).

6.4 Skeletonization

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

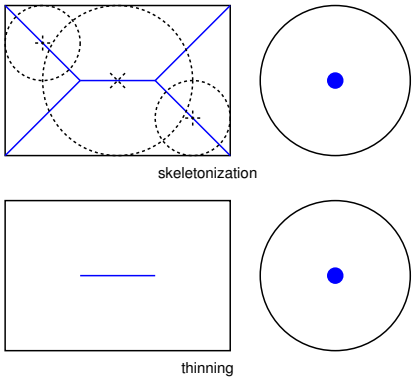


Figure 84: Skeletonization

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

Skeletonization can be implemented with a two-pass conditional erosion, as with thinning. The rule for deleting pixels, however, is slightly different (see figure 84).

6.5 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.

6.6 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.

6.7 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. 85).

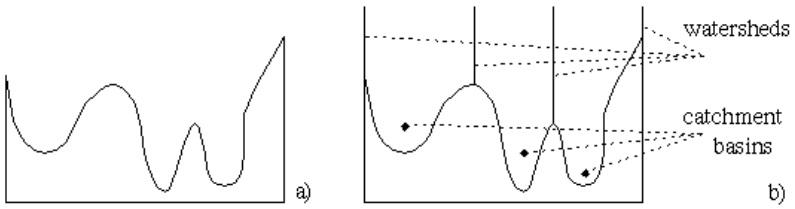


Figure 85: Grundprinzipien: Wasserscheiden und Auffangbecken

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. 86 für eine Visualisierung.

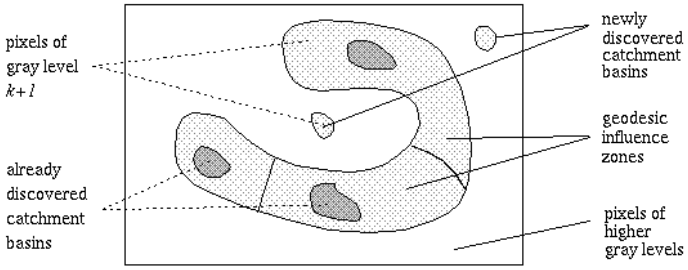


Figure 86: Einflusszonen der Auffangbecken

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. 87 stellt den gesamten Vorgang dar.

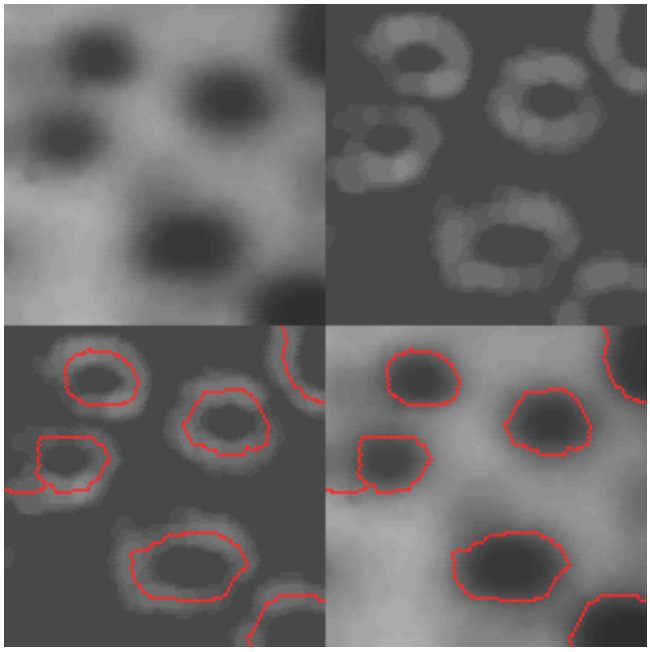


Figure 87: Original, Grandientenbild, Watersheds, original mit Watersheds

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. 88 und 90.c), da es oft eine zu hohe Anzahl an Minima gibt (z.B. durch Verrauschung).

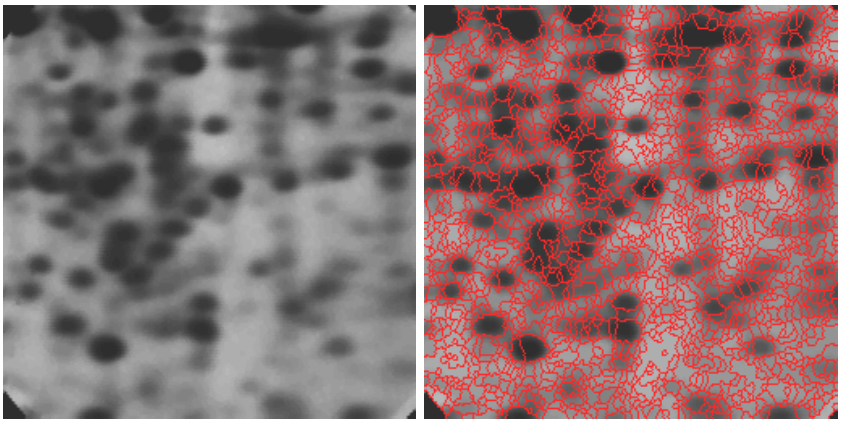


Figure 88: Übersegmentierung

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

- 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. 89 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. 89 rechts und 90.d zeigt entsprechende Ergebnisse. Zusätzlich kann die Anzahl der inneren Marker beschränkt werden oder iene Mindestgrösse verlangt werden.

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. 91). 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.

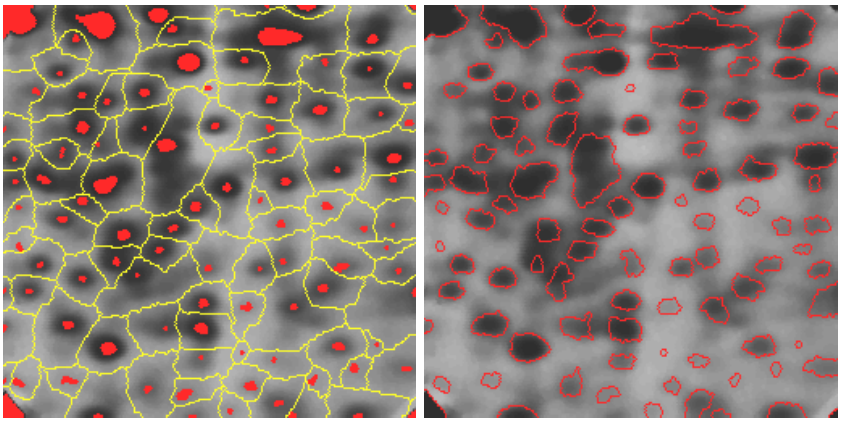


Figure 89: Watersheds mit inneren und äusseren Markern

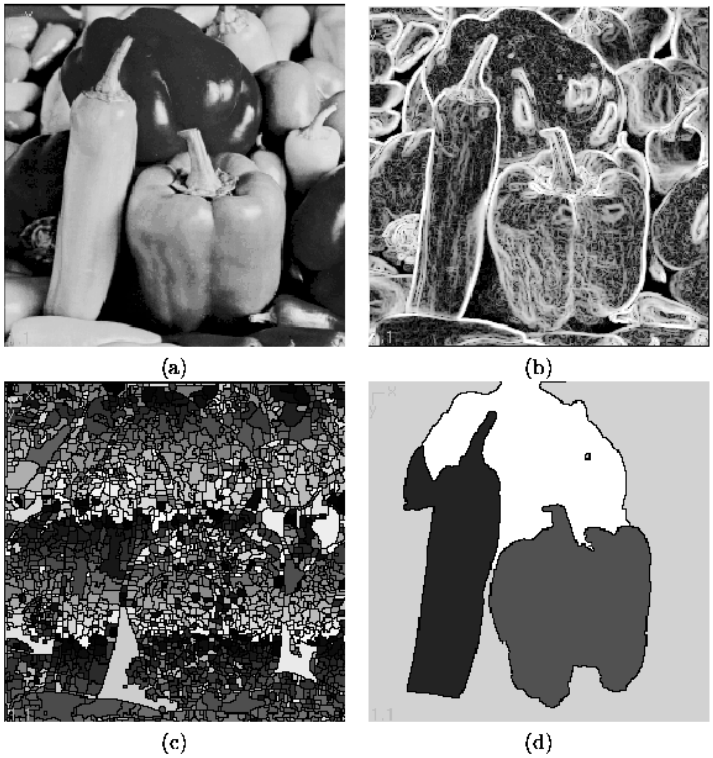


Figure 90: Beispiel: alle Varianten

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

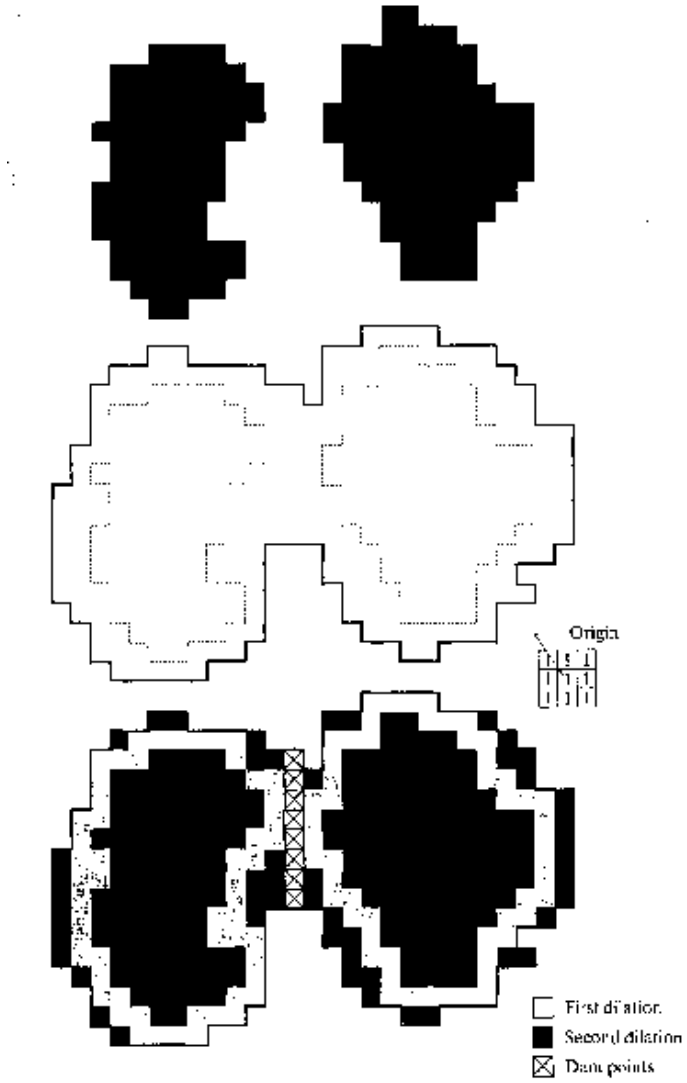


Figure 91: Dammkonstruktion mit Dilation

7 Additional Topics

- Image Forensics
- (Texture) Classification
- Interpolation Techniques
- Noise
- Statistische BVA
- .....