Stoffzusammenfassung zur Bild- und Signalverarbeitung

Jan Krieger

23. September 2006

'The time has come,' the Walrus said,
'To talk of many things:
Of shoes – and ships – and sealingwax –
Of cabbages – and kings –
And why the sea is boiling hot –
And whether pigs have wings.'

– Tweedldee in Through the Looking-Glass by Lewis Carrol

Inhaltsverzeichnis

I Einführung 7

1 Einleitung 8
1.1 Aufbau und Motivation diese Skriptes . . . 8

2 Grundlagen der Bildverarbeitung 9
2.1 Bilddaten . . . 9
2.2 Bildrepräsentation . . . 9
2.3 Gamma-Korrektur . . . 10
2.4 Farbbilder . . . 10
2.5 Diskrete Bilder . . . 11

3 Fourier-Darstellung von Bildern 13
3.1 Sinus- und Cosinus-Schwingungen . . . 13
3.1.1 Geometrische 1D-Repräsentation . . . 13
3.1.2 Geometrische 2D-Repräsentation . . . 14
3.2 Fourier-Reihen . . . 16
3.3 Fourier-Transformation . . . 18
3.4 Diskrete Fourier-Transformation (DFT) . . . 19
3.5 Mehrdimensionale Fourier-Transformationen . . . 21
3.6 Eigenschaften der Fourier-Transformation . . . 22
3.6.1 Allgemeine Eigenschaften . . . 22
3.6.2 Spezielle Eigenschaften der diskreten Transformationen . . . 26
3.7 DFT als diskrete unitäre Transformation und weitere Transformationen . . . 26
3.8 Lokale Fourier-Transformation . . . 27

4 Digitalisierung, Abtastung und Quantisierung 28
4.1 Bilderzeugung, Abtastung . . . 29
4.2 Quantisierung . . . 32

5 Bildrepräsentation 34
5.1 Lauflängenkodierung . . . 34
5.2 Quadtree/Octree . . . 34
5.3 Weitere Kodierungsmethoden . . . 35

II Grundlegende Operationen auf Bilddaten 36

6 Punktoperationen und geometrische Operationen 37
6.1 Punktoperationen . . . 37
6.2 Beispiele für Punktoperationen . . . 38
6.3 Geometrische Transformationen . . . 41

7 Nachbarschaftsoperationen 42
7.1 lineare verschiebungsinvariante Filter . . . 42

7.1.1 Faltungen . . . 42
7.1.2 Eigenschaften von LSI-Filtern . . . 44
7.2 Rangordnungsfilter . . . 47
7.3 rekursive Filter . . . 48

8 Multiskalenrepräsentation 50
8.1 Pyramiden-Zerlegungen . . . 50
8.1.1 Gauß-Pyramide . . . 50
8.1.2 Laplace-Pyramide . . . 51
8.1.3 Richtungszerlegungen . . . 52
8.2 Skalenraumtheorie . . . 52
8.2.1 Diffusionsgleichungen . . . 53
8.2.2 Eigenschaften des Skalenraumes . . . 54

III Einfache Merkmalsextraktion 55

9 Mittelung 56
9.1 Eigenschaften . . . 56
9.2 Rechteckfilter . . . 57
9.3 Binomialfilter . . . 58
9.4 Schnelle großräumige Mittelung . . . 61
9.4.1 Mehrschrittmittelung . . . 61
9.4.2 Rekursive Mittelung . . . 61
9.5 nichtlineare und steuerbare Mittelung . . . 62
9.5.1 Medianfilter . . . 62
9.5.2 gewichtete und einstellbare Mittelung . . . 62

10 Kantendetektion 64
10.1 differentielle Merkmalsbeschreibung . . . 64
10.2 Ableitungsoperatoren . . . 64
10.3 Eigenschaften . . . 65
10.4 Gradientenbasierte Ableitungsfilter . . . 66
10.5 Laplace-Filter . . . 68
10.6 Optimierung der Kantenfilter . . . 70
10.6.1 Allgemeine Optimierungsansätze . . . 70
10.6.2 Regularisierte Kantendetektoren . . . 70

11 Einfache Nachbarschaften 73
11.1 Einführung . . . 73
11.2 Vektordarstellung . . . 74
11.3 Tensordarstellung erster Ordnung . . . 75
11.3.1 Strukturtensor . . . 75
11.3.2 Andere Tensordarstellungen . . . 77
11.4 Lokale Wellenzahl und Phase . . . 78
11.4.1 Hilbertfilter und Hilberttransformation . . . 79
11.4.2 Analytisches Signal . . . 80
11.4.3 Quadraturfilter . . . 81

12 Textur 83
12.1 Einleitung . . . 83
12.2 Statistik erster Ordnung . . . 84
12.3 Orientierung und Textur . . . 85
12.4 Pyramidale Texturanalyse . . . 85

13 Bewegungserkennung **86**
13.1 Grundlagen . . . 86
13.2 Einfache Bewegungsdetektion . . . 87
13.3 Bewegung im Orts-Zeit- und Fourier-Raum . . . 88
13.3.1 Orts-Zeit-Raum . . . 88
13.3.2 Fourier-Raum . . . 89
13.4 Optischer Fluss und Kontinuitätsgleichung . . . 90
13.5 Gradientenmethode . . . 91
13.5.1 Tensormethode . . . 92
13.6 Korrelationsmethode . . . 93
13.7 Phasenmethode . . . 94

14 Inverse Filterung **95**

IV Bildanalyse **97**

15 Segmentierung **98**
15.1 Einleitung . . . 98
15.2 Pixelorientierte Methoden . . . 98
15.3 Kantenbasierte Segmentierung . . . 100
15.4 Regionenorientierte Verfahren . . . 101
15.5 Modellbasierte Segmentierung . . . 103
15.5.1 Regularisierung und Modellierung . . . 103
15.5.2 Hough-Transformation . . . 103
15.5.3 Kontinuierliche Modellierung: Variationsmethode . . . 104
15.5.4 Diffusionsmodelle . . . 105

16 Morphologie **107**
16.1 Dilatation und Erosion . . . 107
16.2 Allgemeine morphologische Operationen . . . 108
16.3 Zusammengesetzte morphologische Operationen . . . 109
16.3.1 Öffnen und Schließen . . . 109
16.3.2 Hit-Miss-Operator . . . 110
16.4 Extraktion von Rändern . . . 111

17 Formrepräsentation **112**
17.1 Momentbasierte Formmerkmale . . . 112
17.2 Richtungsketten . . . 113
17.3 Fourierdeskriptoren . . . 114
17.4 Formparameter . . . 115

18 Klassifikation **117**
18.1 Problemstellung . . . 117
18.2 Einfache Klassifikationsverfahren . . . 119
18.2.1 Nachschaumethode . . . 120
18.2.2 Quadermethode . . . 120
18.2.3 Methode des geringsten Abstandes . . . 120
18.2.4 Methode der höchsten Wahrscheinlichkeit . . . 120

V Mathematische Formelsammlung **121**

19 Komplexe Zahlen **122**

Abbildungsverzeichnis 125

A Weblinks 128

Literaturverzeichnis 129

Teil I

Einführung

1 Einleitung

1.1 Aufbau und Motivation diese Skriptes

Diese Skript ist als Vorbereitung auf meine Diplomprüfung über die Vorlesung Bildverarbeitung bei Prof. Jähne (IWR, Uni Heidelberg), wie ich sie im WS 05/06 gehört habe entstanden. Ich habe vornehmlich den Inhalt dieser Vorlesung zusammengefasst, die weitgehend dem Buch [Jähne 2005]. Ich hoffe die Darstellung in diesem Skript kann einigen Helfen, sich in das Gebiet einzuarbeiten, oder sich auf eine Prüfung vorzubereiten.

Im ersten Abschnitt werden zunächst einmal mathematische Grundlagen, wie etwa die Fouriertransformation diskutiert.

Üblicherweise gliedert sich die Klassifizierung der Inhalte eines Bildes in mehrere Schritte, die in Abb. 1.1 dargestellt sind und im Detail in den einzelnen Abschnitten abgehandelt werden. Es muss nicht für alle Anwendungen der volle Weg gegangen werden. In vielen Fällen kann man etwa schon vorher abbrechen (z.B. nach der Objektbeschreibung etz.)

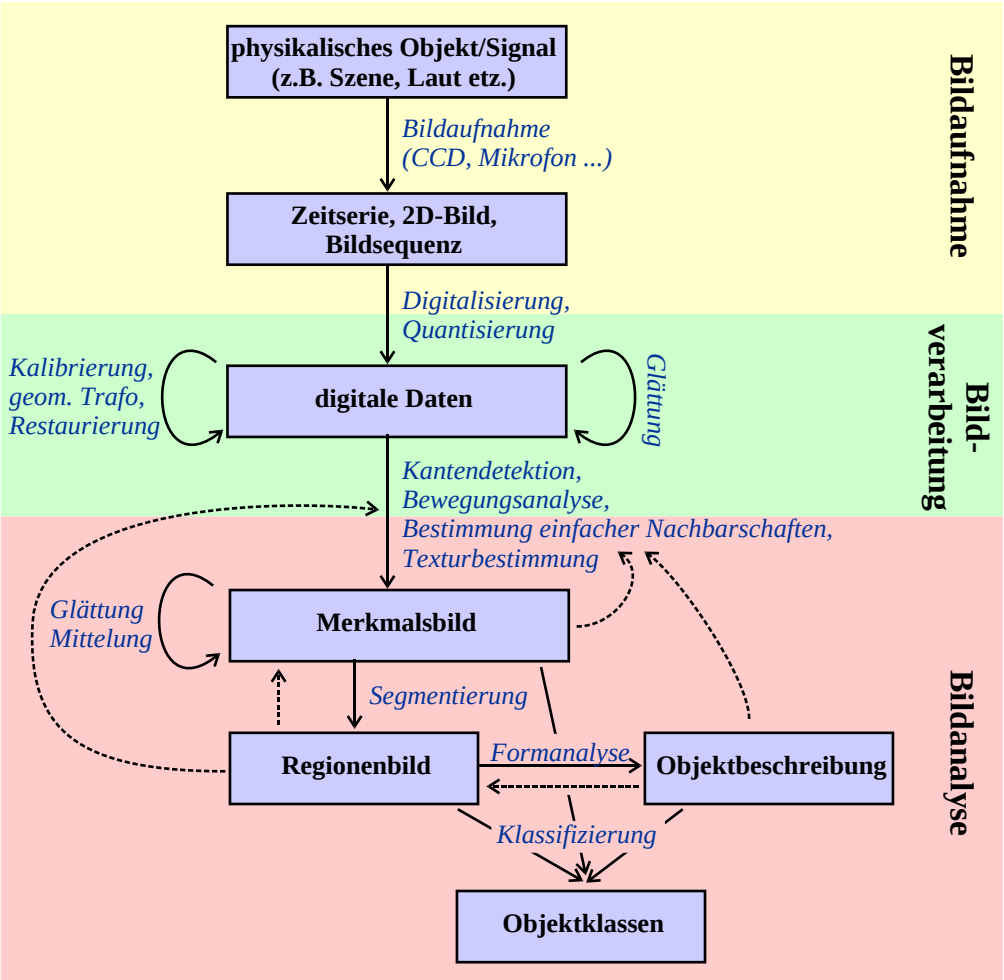


Abb. 1.1: Übersicht über die einzelnen Schritte einer Klassifizierung in der Bildverarbeitung

2 Grundlagen der Bildverarbeitung

2.1 Bilddaten

Zunächst denkt man bei Bildverarbeitung vor Allem an die computergestützte Verarbeitung von Kamerabildern. Der Titel dieses Skriptes wurde aber schon etwas allgemeiner gewählt. So können die selben Techniken, die für die Verarbeitung von Kameraaufnahmen angewendet werden auch auf höher dimensionale Daten angewendet werden. So werden hier etwa auch Sequenzen von Bildern betrachtet, die als drei-dimensionale Bilder aufgefasst werden können. Genauso könnte man sich vorstellen eine Sequenz von drei-dimensionalen Bildern zu verarbeiten (insg. 4-dimensional). Etwas allgemeiner, kann man beliebig dimensionale Messdaten verarbeiten. Ein Beispiel wäre etwa die Daten, die eine Wetterstation zu verschiedenen Zeitpunkten misst (z.B. Temperatur innen/außen, Luftdruck, Luftfeuchte, Regenmenge, Windgeschwindigkeit). Dies ergäbe etwa schon ein sieben-dimensionales „Bild“.

2.2 Bildrepräsentation

Eine übliche Kamera liefert ein Bild mit $N \times M$ **Bildpunkten** (**Pixeln**). Jedem dieser Bildpunkte $\vec{x} = (x, y)$ ist ein **Grauwert** zugeordnet. Man kann also ein Bild als Funktion der Variablen x, y auffassen:

$$g(\vec{x}) : \vec{R}^n \quad \rightarrow \quad \vec{R}$$

Hier gibt n die Dimensionalität der Daten an. $n = 1$ steht für ein Zeilenbild, $n = 2$ für ein „normale“ Bild, $n = 3$ für ein Volumenbild, wie es etwa in der Medizin als Ausgabe eine Computertomographen auftaucht, oder für eine Sequenz von 2D-Bildern usw.

Übliche Computerbilder liegen im Zahlenbereich $g(\vec{x}) \in [0..255 = 2^8 - 1]$ oder $g(\vec{x}) \in [0..65535 = 2^{16} - 1]$. Führt man punktweise, arithemtische Operationen mit Bilder aus, wie etwa die Subtraktion

$$g'(\vec{x}) = f(\vec{y}) - g(\vec{y})$$

so ändert sich dieser Zahlenbereich. So könnte etwa das Ergebnis einer Subtraktion zweier Byte-Bilder (aus $[0..255]$) den minimalen Wert $0 - 255 = -255$ und den maximalen Wert $255 - 0 = 255$ haben. Damit gilt dann $g'(\vec{x}) \in [-255..255]$. Dieser Bereich ist mit einem Byte nicht mehr darstellbar, sodass man eine der folgenden Operationen ausführen muss:

- Erweiterung des Datentyps: `unsigned byte` $\rightarrow$ `signed int`
- Kompression des Datenbereiches und Typkonversion: $g'(\vec{x}) \quad \rightarrow \quad g'(\vec{x})/2 \in [-127..127]$ und `unsigned byte` $\rightarrow$ `signed byte`
- Verschiebung der Werte und Typkonversion: $g'(\vec{x}) \quad \rightarrow \quad g'(\vec{x}) + 255 \in [0..511]$ und `unsigned byte` $\rightarrow$ `unsigned int`

Man kann die obigen Operationen auch kombinieren. So kann man etwa den Wertebereich verschieben und danach verkleinern, also

$$g'(\vec{x}) \quad \rightarrow \quad \big(g'(\vec{x}) + 255\big)/2 \in [0..255]$$

Dann ist keine Anpassung des Wertebereiches nötig. Das einfachste ist aber wohl mit genügend großen Datentypen zu rechnen. Man kann Bilder natürlich auch als Gleitkommazahl darstellen. Dabei ist aber

zu beachten, dass Gleitkommarechnungen viel langsamer sind, als Integer-Rechnungen. Spätestens bei der Fourier-Transformation wird man aber nicht mehr um Gleitkomma-Bilder herumkommen, die dann evtl. auch noch komplex ($g(\vec{x}) \in \mathbb{C}$) sein müssen.

In vielen Bildverarbeitungssystemen sind spezielle Operationen implementiert, die als **Sättigungsarithmetik** bezeichnet werden. Dabei ergibt $0-n = 0$ und $255+n = 255$, $n \in \mathbb{N}$. Die Ergebnisse können also nicht größer als die obere und kleiner als die untere Schranke des Wertebereiches eines Pixels werden. Altervativ kann man auch **Moduloarithmetik** verwenden, bei der etwa $a +_m b := (a + b) \mod 256$ gilt. Überschreitet das Ergebnis den Wertebereich, so wird es von unten in den Wertebereich projiziert. So ist etwa $240 +_m 30 = 270 \mod 256 = 14$. Dies führt aber dazu, dass überlaufende Rechnungen, die eigentlich zu sehr „hellen“ Werten führen sollten auf „dunkle“ Werte abgebildet werden.

2.3 Gamma-Korrektur

Das menschliche Auge hat ein logarithmisches Helligkeitsempfinden. Dies bedeutet, dass kleine Unterschiede bei niedrigen Intensitäten schwächer wahrgenommen werden, als kleine Unterschiede bei hohen Intensitäten. Die meisten Kameras haben aber eine lineare Charakteristik, sodass evtl. eine Korrektur nötig ist. Diese erfolgt so:

$$g(\vec{x}) \quad \rightarrow \quad \left[g(\vec{x})\right]^{\gamma}$$

Durch Variation des **γ-Wertes** kann man somit den Kontrastumfang (Fähigkeit, Intensitätsunterschiede wahrzunehmen) erhöhen. Für eine lineare Darstellung wählt man $\gamma = 1$.

2.4 Farbbilder

Behandelt man Farbbilder, so muss der Farbwert an einem Bildpunkt dargestellt werden. Dazu gibt es verschiedene **Farbsysteme**. Einige gebräuchliche sind:

- **RGB:** Jede Farbe wird durch additive Mischung aus den Farben rot, grün und blau zusammengesetzt. Zu jedem Bildpunkt wird also ein 3-Tupel (r, g, b) gespeichert. Die Werte können üblicherweise im Bereich zwischen 0 und 255 (ein Byte je Wert), oder $[0..1]$ liegen. Die Werte geben dabei die Intensität der jeweiligen Grundfarbe an. So erhält man beispielsweise die Farben in Abb. 2.1.

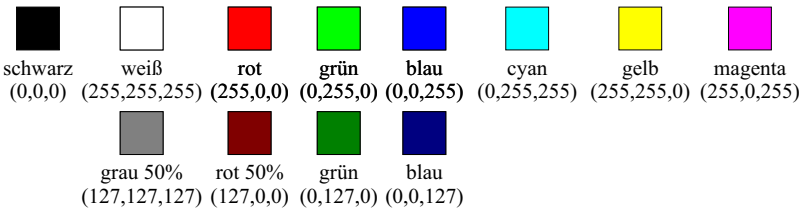


Abb. 2.1: Beispiele für RGB-Farben

  Übliche BIldschirme und Grafikkarten für PCs basieren auf diesem Farbmodell.

- **HSB:** Hier werden ebenfalls drei Werte abgespeichert. Der erste Wert h gibt die Farbe an (Winkel im Frabkreis), wobei 360 der Farbe rot entspricht, 120 ist grün und 240 blau. Die restlichen Farben liegen dazwischen, etwa gelb bei 60 und magenta bei 300. Der zweite Wert s gibt die Sättigung und der dritte Wert b die Helligkeit der Farbe an.
- **CMYK:** Dieses Modell wird im Offset-Druck eingesetzt, wo man Bilder aus den Farben Cyan (c), Magenta (m), gelb (y) und schwarz (k) zusammensetzt. Viele Bildbearbeitugsprogramm unterstützen dieses Farbformat nicht. Es wird aber auch von „normalen“ Tintenstrahl- und Laser-Farbdruckern eingesetzt.

2.5 Diskrete Bilder

Die obige Definition bedeutet ein kontinuierliche Funktion. Oft setzt sich aber ein Bild nur aus einer endlichen Zahl von Bildpunkten zusammen (z.B. $n \times m$ Pixel). Dann sind die Komponenten von $\vec{x}$ nicht mehr aus $\mathbb{R}$, sondern aus $\mathbb{N}$. Ein Bild entspricht dann einer Rechteckmatrix:

$$\begin{array}{|c|c|c|c|}
\hline
g_{0,0} & g_{0,1} & \cdots & g_{0,N-1} \\
\hline
g_{1,0} & g_{1,1} & \cdots & g_{1,N-1} \\
\hline
\vdots & \vdots & \ddots & \vdots \\
\hline
g_{M-1,0} & g_{M-1,1} & \cdots & g_{M-1,N-1} \\
\hline
\end{array}$$

Man stellt dann die Einzelnen Pixel eines $M \times N$-Bildes so dar:

$$g_{m,n} = g_{\text{Zeile,Spalte}}, \qquad m \in [0..M-1], n \in [0..N-1]$$

Der erste Index steht also für die Zeile, der zweite für die Spalte.

Bisher wurde die Pixel als rechteckig angenommen. Dies ist auch in den meisten Fällen so. Man kann aber eine Fläche auch mit drei- und sechseckigen Pixeln füllen (siehe Abb. 2.2)

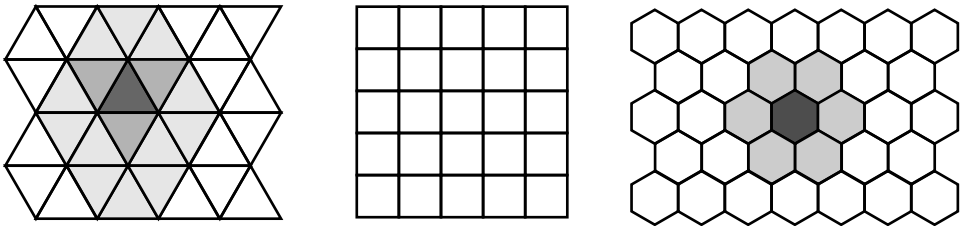
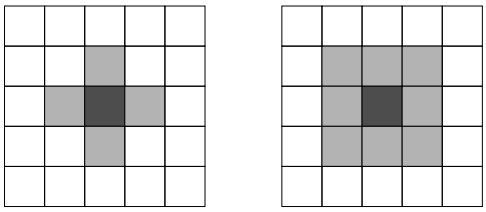


Abb. 2.2: verschiedene raumfüllende 2D-Gitter

Solche Pixelanordnungen machen aber eher in Simulationen Sinn (z.B. zelluläre Automaten), in denen die Form des Gitters die Form der entstehenden Muster beeinflussen kann. SO haben Spiralen auf den unterschiedlichen Gittern unterschiedliches Aussehen. In höher dimensionalen Räumen werden die Formen komplizierten. Im eindimensionalen kann man nur regelmäßige Intervalle definieren. In der Bildverarbeitung spielen aber nur Rechteckgitter eine Rolle.

In der obigen Abbildung sind verschiedene **Nachbarschaftsbeziehungen** eingezeichnet. Diese geben Auskunft darüber, welche der angrenzenden Pixel als direkt benachbart angesehen werden. Im zweidimensionalen gibt es zwei grundlegende Nachbarschaften. Sie sind in Abb. 2.3 dargestellt.



von-Neumann Nachbarschaft Moore Nachbarschaft

Abb. 2.3: 2D-Nachbarschaftsbeziehungen

Im dreidimensionalen kann man ähnliche Nachbarschaften definieren.

3 Fourier-Darstellung von Bildern

3.1 Sinus- und Cosinus-Schwingungen

Im folgenden werden häufig die Sinus- und Cosinusfunktionen (**harmonische Funktionen**) benötigt. Deswegen sollen hier kurz einige wichtige EIgenscgaften zusammengestellt werden:

Beziehung zwischen Sinus und Cosinus: $\cos(\alpha) = \sin(\alpha + \pi/2) = \sin(\alpha + 90°)$

3.1.1 Geometrische 1D-Repräsentation

In Abb. 3.1 sind eindimensionale Sinus- und Cosinus-Schwingungen gezeigt.

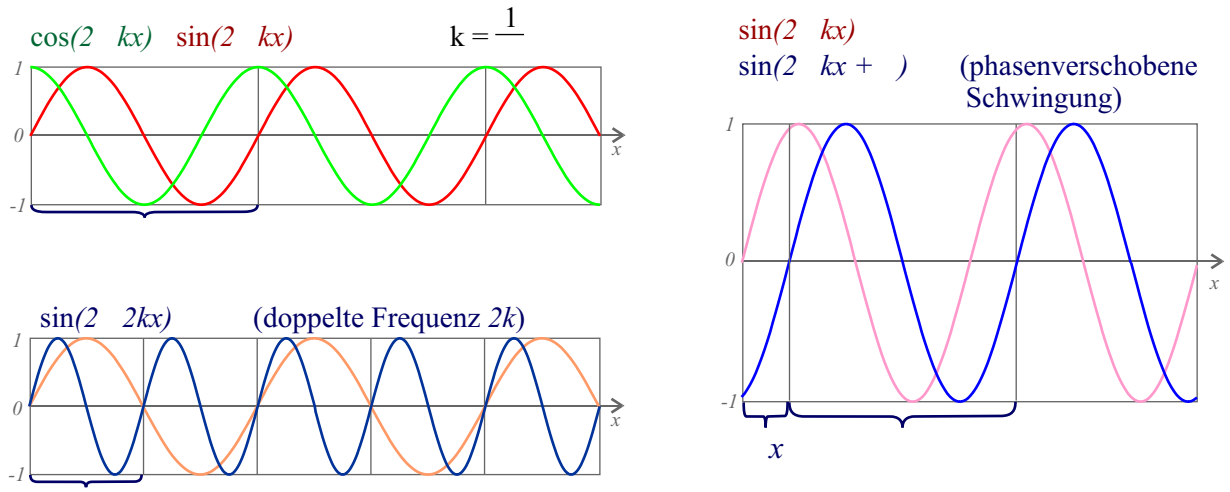


Abb. 3.1: 1D-Sinus- und Cosinus-Schwingungen, mit unterschiedlicher Phase und Frequenz.

Es zeigt sich, dass diese Funktionen eine typische Periodizität aufweisen. Nach dem Abstand λ (**Wellenlänge**) auf der x-Achse wiederholt sich das Muster wieder. Die Funktionen $\sin(2\pi \cdot kx)$ und $\cos(2\pi \cdot kx)$ hängen neben x auch noch von einem Parameter k (**Frequenz**) ab. Man definiert die **Wellenzahl** oder **räumliche Frequenz** k:

$$k = \frac{1}{\lambda} = \text{Schwingungsperioden im Einheitsintervall } [0..1]$$

Manchmal (vor allem in der Physik) trifft man auch die Definition $k = \frac{2\pi}{\lambda}$ an. Dann ist der Faktor 2π in der Sinus- oder Cosinus-Funktion nicht nötig.

Die Bezeichnung räumliche Frequenz stammt von der Analog zu zeitlichen Schwingungen. Dort nennt man die Periode der Schwingung (=Wellenlänge) T und die Anzahl der Schwingungen pro Einheitszeitintervall (z.B. $1\,\text{s} \;\Rightarrow\; [\nu] = 1\,\text{Hz}$) Frequenz $\nu = \frac{1}{T}$, oder Kreisfrequenz $\omega = \frac{2\pi}{\lambda}$.

In der obigen Abbildung ist rechts auch eine Verschobene Schwingung gezeigt. Um eine solche Verschiebung zu erreichen, muss man eine Konstante φ (**Phase**) zum Argument des Sinus addieren. Dieses φ hängt mit der Verschiebung Δx so zusammen:

$$\varphi = \frac{\Delta x}{\lambda} \cdot 2\pi$$

Die Verschiebung wird also im Bogenmaß $[0..2\pi]$ oder im Gradmaß $[0..360°]$ gemessen.

Jede harmonische 1D-Schwingung lässt sich also durch Angabe von zwei Parametern, nämlich der Frequenz k und der Phase φ genau bestimmen. Zusätzlich kann man noch einen reellen Vorfaktor r (**Amplitude**) hinzufügen, der die Schwingung skaliert:

$$f_{\text{harmonisch}}(x) = r \cdot \cos(2\pi \cdot kx - \varphi)$$

komplexe Repräsentation, 1D: Oft benötigt man zwar nur reelle Sinus- oder Cosinusschwingungen. Um aber z.B. eine Phase φ zu einer Grundschwingung $\cos(2\pi kx)$ hinzuzufügen ist eine nicht-triviale Rechnung mit Additionstheoremen nötig. Dies kann man umgehen, wenn man die Beziehung

$$z = a + \mathrm{i}b = |z| \cdot \mathrm{e}^{\mathrm{i}\phi} = |z| \cdot \big(\cos(\phi) + \mathrm{i}\sin(\phi)\big), \qquad i \in \mathbb{C}, \;\; a, b, \phi, |z| \in \mathbb{R}$$

für beliebige komplexe Zahlen ausnutzt. Man kann dann obige Schwingung so schreiben:

$$r \cdot \cos(2\pi kx) = \mathrm{Re}\left\{r \cdot \mathrm{e}^{\mathrm{i}\cdot 2\pi \cdot kx}\right\}$$

Man kann also mit der komplexen Schwingung $r \cdot \mathrm{e}^{\mathrm{i}\cdot 2\pi \cdot kx}$ rechnen und hernach nur den Realteil verwenden. Dies hat den Sinn, dass man jetzt eine Phase φ durch einfache Multiplikation mit $\mathrm{e}^{-\mathrm{i}\varphi}$ hinzufügen kann:

$$\mathrm{Re}\left\{r \cdot \mathrm{e}^{\mathrm{i}\cdot 2\pi \cdot kx} \cdot \mathrm{e}^{-\mathrm{i}\varphi}\right\} = \mathrm{Re}\left\{r \cdot \mathrm{e}^{\mathrm{i}\cdot(2\pi \cdot kx - \varphi)}\right\} = r \cdot \cos(2\pi kx - \varphi)$$

3.1.2 Geometrische 2D-Repräsentation

Auch in 2D-Bildern kann man harmonische, periodische Strukturen haben, die sich dann in Grauwertschwankung äußern, wie sie in der folgenden Abb. 3.2 gezeigt sind.

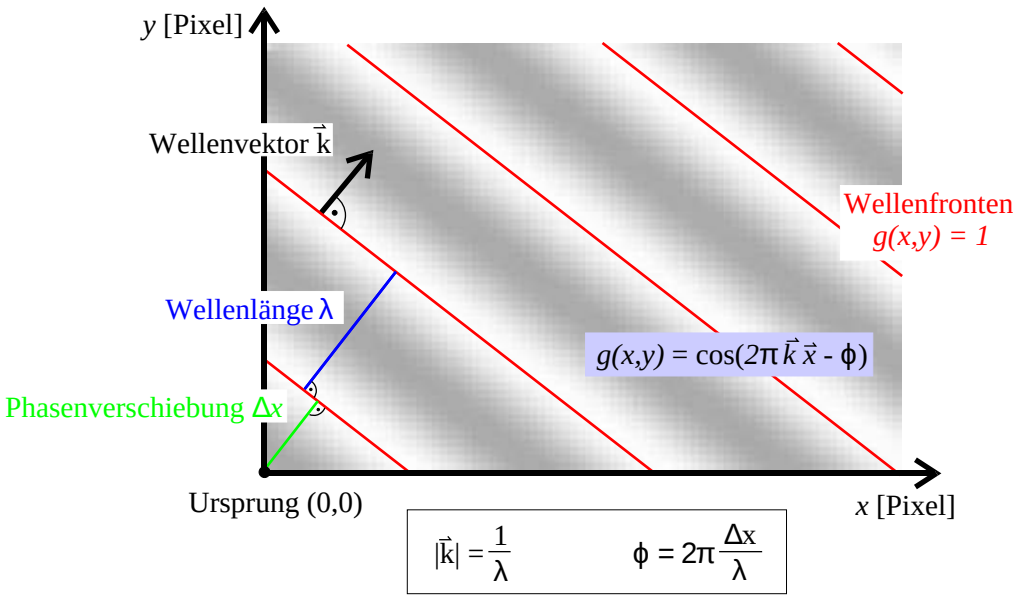


Abb. 3.2: zur Definition des 2D-Wellenvektors

Analog zum obigen Fall kann man hier auch wieder Parameter für die Welle angeben. Das Bild wird von einer Funktion

$$g(\vec{x}) = r \cdot \cos\big(2\pi \cdot \vec{k} \cdot \vec{x} - \varphi\big)$$

erzeugt. Darin kommen die Parameter r (Amplitude), φ (Phase) und $\vec{k} = \begin{pmatrix} k_x \\ k_y \end{pmatrix}$ (**Wellenvektor**) vor. Die ersten beiden Parameter sind analog zum 1D-Fall. Der Wellenvektor $\vec{k}$ entspricht der Wellenzahl/räumlichen Frequenz k und ist betragsmäßig gleich dieser. Zusätzlich steht er senkrecht auf den Wellenfronten der periodischen Struktur. Die Wellenlänge ist jetzt als der senkrechte Abstand zwischen zwei Wellenfronten definiert. Eine solche 2D-Struktur wird also durch den Satz der vier Parameter $\{r, \varphi, k_x, k_y\}$ vollständig beschrieben.

Auch hier ist es wieder sinnvoll eine komplexe Repräsentation einzuführen:

$$g(\vec{x}) = \mathrm{Re}\left\{r \cdot \mathrm{e}^{\mathrm{i}\cdot(2\pi\cdot\vec{k}\cdot\vec{x}-\varphi)}\right\}$$

In den folgenden Abschnitten wird die Notation aus diesem Abschnitt verwendet. Das Symbol k stellt also immer eine räumliche Frequenz (Wellenzahl) dar.

3.2 Fourier-Reihen

Bisher wurden Bilder so dargestellt, dass zu jedem Bildpunkt ein Grau (oder Farb-)wert gespeichert wurde. Nun betrachte man aber die Strukturen in Abb. 3.3.

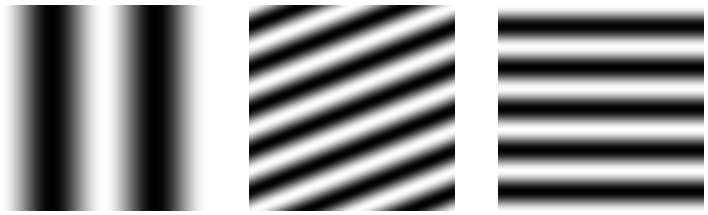


Abb. 3.3: Wellenstrukturen in 2D-Bildern, mit verschiedenen Frequenzen und Ausrichtungen

In ihnen steckt weit weniger Information, als der Grauwert an jedem Bildpunkt. Sie stellen einfach Sinusförmige (und gekippte) Schwankungen des Grauwertes dar. Es würde also zu ihrer Abspeicherung genügen die Frequenz dieser Grauwertvariationen (Schwingungen) zu kennen, sowie ihre Verkippung. Dies bringt einen auf die Idee der Fourier-Darstellung von Bildern. Dazu benutzt man eine Verallgemeinerung des folgenden Satzes aus der Analysis:

Satz 3.1 (FOURIER-Entwicklung in e^{ikx}) *Für eine beliebige Funktion $f \in \mathcal{R}[0, 2\pi]$ definiert man die zugehörige Fourier-Summe durch:*

$$F_n^f(x) := \sum_{k=-n}^{n} c_k \, \mathrm{e}^{ikx} \qquad \textit{mit} \qquad c_k := \frac{1}{2\pi} \int_0^{2\pi} f(x) \, \mathrm{e}^{-ikx} \, \mathrm{d}x; \quad k \in \mathbb{Z}.$$

Im Falle der Konvergenz heißt der Limes der Fourier-Summen, die Fourier-Reihe:

$$F_\infty^f(x) := \sum_{k=-\infty}^{\infty} c_k \, \mathrm{e}^{ikx} = \lim_{n\to\infty} \sum_{k=-n}^{n} c_k \, \mathrm{e}^{ikx} \, .$$

Sei nun weiter $f \in \mathcal{R}[0, 2\pi]$ eine 2π-periodische Funktion. Dann konvergiert ihre Fourier-Reihe im quadratischen Mittel gegen f und mit ihren Fourier-Koeffizienten c_k gilt die sog. Vollständigkeitsrelation:

$$2\pi \sum_{k=-\infty}^{\infty} |c_k|^2 = \|f\|^2$$

Dieser Satz besagt, dass sich beliebige 2π-periodische Funktionen in komplexe e-Funktionen entwickeln lässt. Diese Funktionen leben auf einem Vektorraum $\mathcal{R}[0, 2\pi]$ quadratintegrabler, 2π-periodischer Funktionen. Für dieses gelten die üblichen Regeln der linearen Algebra, sodass man auch eine Basis finden kann, in der beliebige Elemente (Vektoren) dieses Raumes dargestellt werden können. Dazu zunächst ein kleiner Exkurs:

Basiswechsel im $\mathbb{R}^2$: Man betrachte hier den zweidimensionalen Vektorraum $\mathbb{R}^2$. Vektoren dieses Raumes lassen sich als Spaltenvektoren schreiben, z.B.:

$$\vec{a} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \quad \vec{e}_x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \vec{e}_y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

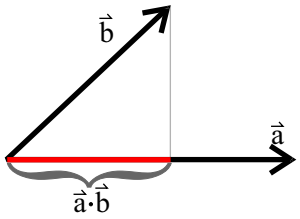
Zwei linear unabhängige Vektoren dieses Raumes bilden nun eine sog. Basis. Dies können beispielsweise $\vec{e}_x$ und $\vec{e}_y$ sein (kanonische Basis). Man kann nun jeden Vektor des Raumes als Linearkombination dieser Vektoren darstellen:

$$\vec{a} = a_x \cdot \vec{e}_x + a_y \cdot \vec{e}_y = 1 \cdot \vec{e}_x + 1 \cdot \vec{e}_y, \quad \vec{b} = b_x \dot{\vec{e}}_x + b_y \cdot \vec{e}_y = -1 \cdot \vec{e}_x + 2 \cdot \vec{e}_y$$

Um die Koeffizienten $a_{x/y}, b_{x/y}$ vor den Basisvektoren zu berechnen kann man, das sog. Skalarprodukt auf diesem Raum verwenden. Das Skalarprodukt ist hier folgendermaßen definiert:

$$\langle \vec{a}, \vec{b} \rangle \equiv \vec{a} \cdot \vec{b} := a_x b_x + a_y b_y$$

Dieses Produkt berechnet den Anteil des Vektors $\vec{b}$, der parallel zu $\vec{a}$ liegt:



Stehen zwei Vektoren senkrecht aufeinander, so ist ihr Skalarprodukt 0. Die Koeffizienten $c_{x/y}$ eines beliebigen Vektors $\vec{c}$ ergeben sich dann so

$$c_x = \langle \vec{c}, \vec{e}_x \rangle, \quad c_y = \langle \vec{c}, \vec{e}_y \rangle$$

Man könnte also auch eine andere Basis wählen (z.B. $\{\vec{a}, \vec{b}\}$). Dort gilt dann etwa für $\vec{d} = \vec{e}_x = (1, 0)^t$:

$$d_x = \langle \vec{d}, \vec{a} \rangle = 1, \quad d_y = \langle \vec{d}, \vec{b} \rangle = -1$$

Auf dem Vektorraum $\mathcal{R}[0, 2\pi]$ definiert man nun folgendes Skalarprodukt:

$$\langle f, g \rangle := \int_0^{2\pi} f(x) \cdot g^*(x) \, \mathrm{d}x$$

Die Funktionen f, g sind Elemente des Vektorraumes und $g^*(x)$ bedeutet das komplex-konjugierte der Funktion $g(x)$. Mit diesem Skalarprodukt wird aus obiger Formel für die Fourier-Koeffizienten c_k nichts anderes, als

$$c_k = \frac{1}{2\pi} \cdot \langle f, \mathrm{e}^{-ikx} \rangle$$

Man kann die Fourier-Transformation also als Basiswechsel im Raum $\mathcal{R}[0, 2\pi]$ auffassen. Damit lassen sich zu jeder Funktion $f \in \mathcal{R}[0, 2\pi]$ Fourier-Koeffizienten ermitteln, mit denen man f als Linearkombination von Exponentialfunktionen e^{ikx} mit verschiedenen $k \in \mathbb{Z}$ darstellen kann. Man hat also eine neue Beschreibung $\{c_0, c_{\pm 1}, c_{\pm 2}, ...\}$ der selben Funktion f gefunden. Bisher hat man den Wert der Funktion an jeder Stelle x angegeben. Dafür lässt sich ebenfalls eine Basis finden:

$$f(x) = \sum_k c'_k \cdot \delta(x - x_k)$$

Die Dirac-δ-Funktionen $\delta(x - x_k)$ sind überall 0, außer an x_k. Dort streben sie gegen unendlich und es gilt:

$$\int f(x) \cdot \delta(x - x_0) \, \mathrm{d}x = f(x_0)$$

Dabei muss man nur voraussetzen, dass der Integrationsbereich die Stelle x_0 überstreicht. Man stellt also die Funktion als Summe (eigentlich Integral!!!) über die Werte an den einzelnen Stellen x_k dar.

Die folgende Abb. 3.4 zeigt die Approximation einer Sägezahnfunktion durch Fourier-Reihen.

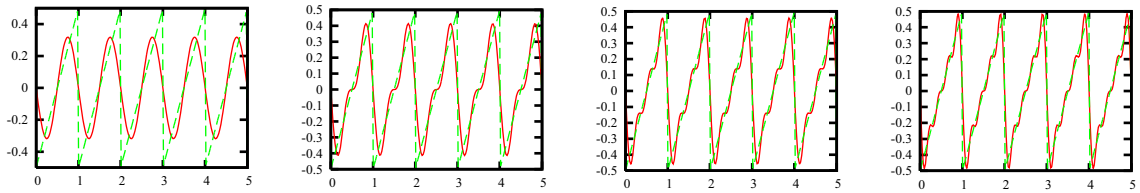


Abb. 3.4: Fourier-Reihendarstellung einer Sägezahnfunktion (grün: Sägezahn, rot: Fourierreihe). Es werden die ersten ein bis vier Koeffizienten berücksichtigt.

Je mehr Koeffizienten also zu Approximation herangezogen werden, desto besser wird diese. Außerdem stellt man fest, dass höhere Frequenzen zu steileren Anstiegen/Abfällen führen.

3.3 Fourier-Transformation

Es zeigt sich, dass man obige Reihenentwicklung auf beliebige Funktionen (die nicht notwendig periodisch sein müssen) erweitern kann. Man kommt dann zur:

Satz 3.2 (FOURIER-Transformation) *Sei $g(x) : \mathbb{R} \to \mathbb{C}$ eine quadratintegrable Funktion, d.h.*

$$\int_{-\infty}^{\infty} |g(x)|^2 \, \mathrm{d}x < \infty.$$

*Dann ist ihre **Fourier-Transformierte** $\hat{g}(k)$ gegeben durch:*

$$\hat{g}(k) = \mathcal{F}[g] = \int_{-\infty}^{\infty} g(x) \cdot \mathrm{e}^{-2\pi \mathrm{i} kx} \, \mathrm{d}x =: \langle g(x), \mathrm{e}^{2\pi \mathrm{i} kx} \rangle$$

Ist nur $\hat{g}(k)$ bekannt, so erhält man $g(x)$ durch Rücktransformation:

$$g(x) = \mathcal{F}^{-1}[\hat{g}] = \int_{-\infty}^{\infty} \hat{g}(k) \cdot \mathrm{e}^{2\pi \mathrm{i} kx} \, \mathrm{d}x$$

Dies ist die kontinuierliche Verallgemeinerung der obigen Entwicklung. Die Funktion $\hat{g}(k)$ übernimmt die Rolle der Koeffizienten c_k und die Summen gehen in Integrale über. Der erste Ausdruck entspricht wieder einem Skalarprodukt auf dem Raum L^2 der quadratintegrablen Funktionen (einzige Voraussetzung!). Der zweite Ausdruck entspricht der Linearkombination der Basisvektoren $\mathrm{e}^{2\pi \mathrm{i} kx}$ (man beachte die komplexe Konjugation im Skalarprodukt, die im ersten Integral $\mathrm{e}^{2\pi \mathrm{i} kx} \to \mathrm{e}^{-2\pi \mathrm{i} kx}$ bewirkt). Beide Ausdrücke sind (bis auf das $-$-Zeichen) vollkommen symmetrisch. Man führt noch folgende Schreibweise für Fourier-Paare $(g, \hat{g})$ ein:

$$g(x) \circ\!\!-\!\!\bullet \; \hat{g}(k)$$

Die folgende Tabelle führt einige Fourier-Paare auf:

Normal-/Realraum $f(x)$		**Fourierraum** $\hat{f}(k)$	**Erklärung**
$\delta(x)$	$\circ\!\!-\!\!\bullet$	1	Die Delta-Funktion enthält alle Frequenzen
$\cos(k_0 x)$	$\circ\!\!-\!\!\bullet$	$\frac{1}{2}\big(\delta(k-k_0)+\delta(k-k_0)\big)$	Der Kosinus enthält nur die Frequenzkomponente k_0
$\sin(k_0 x)$	$\circ\!\!-\!\!\bullet$	$\frac{1}{2}\big(\delta(k-k_0)-\delta(k-k_0)\big)$	
$\mathrm{e}^{-\pi x^2}$	$\circ\!\!-\!\!\bullet$	$\mathrm{e}^{-\pi k^2}$	Gauß transformiert sich in Gauß

Fourier-Transformation, anschaulich: Man kann die Fourier-Transformation so verstehen, dass sie angibt, welche Frequenzanteile wie stark in einer Funktion (in einem Signal) $g(x)$ vorhanden sind. Man kann sich dies an Tönen gut vorstellen. Je höher etwa eine Stimme aus dem Radio klingt, um größer ist der Anteil hoher Frequenzen an dem Signal ($\hat{g}(k)$ ist also groß für große k und klein für kleine k). Am Stellschalter für die höhen kann man diese hohen Anteile dämpfen, sodass eine hohe Stimme tiefer erscheint (Prinzip der Filterung, siehe später). Dagegen enthält z.B. die Basslinie eines Musikstückes sehr viele tiefe Frequenzen ($\hat{g}(k)$ ist also groß für kleine k und klein für große k). Dämpft man nun die Bässe am entsprechenden Stellknopf, so gehen diese im Signal verloren und man nervt seine Mitbewohner nur halb so stark mit Erschütterungen der Wohnung ;-)

Bei der Rücktransformation werden dann die einzelnen Basisschwingungen $\mathrm{e}^{\mathrm{i}kx}$ mit $c(k)$ skaliert und aufintegriert.

3.4 Diskrete Fourier-Transformation (DFT)

Für die praktische Anwendung dieses mathematischen Konzeptes in der Bildverarbeitung muss man eine Transformation finden, die auf diskreten Bildern arbeitet (siehe vorheriger Abschnitt 2.5). Dies leistet die diskrete Fourier-Transformation. Sie geht nun nicht mehr von einer kontinuierlichen Funktion $g(x)$ aus, sondern von einer Menge von N „Messpunkten“ (z.B. die Grauwerte eines Zeilenbildes). Es gilt:

Satz 3.3 (Diskrete Fourier-Transformation (1D-DFT)) *Die 1D-DFT bildet eine geordnete Menge von Werte (N-Tupel)*

$$g = \{g_0, g_1, ..., g_N\}$$

auf eine andere geordnete Menge

$$\hat{g} = \{g_0, g_1, ..., g_N\}$$

ab. Beide lassen sich auch als Vektoren interpretieren. Die Abbildungvorschrift lautet:

$$\hat{g}_k = \frac{1}{N}\sum_{n=0}^{N-1} g_n \cdot \exp\left(-\frac{2\pi \mathrm{i} nk}{N}\right) = \frac{1}{N}\langle g, \exp\left(\frac{2\pi \mathrm{i} nk}{N}\right)\rangle, \quad 0 \le k < N$$

Die Rücktransformation lautet:

$$g_n = \frac{1}{N}\langle \hat{g}, \exp\left(-\frac{2\pi \mathrm{i} nk}{N}\right)\rangle = \sum_{k=0}^{N-1} \hat{g}_k \cdot \exp\left(\frac{2\pi \mathrm{i} nk}{N}\right), \quad 0 \le n < N$$

Dies entspricht der Diskretisierung der kontinuierlichen Fourier-Transformation aus 3.3. Die folgende Abb. 3.5 zeigt für $N = 16$ alle Basisfunktionen in Real- und Imaginärteil aufgespaltet.

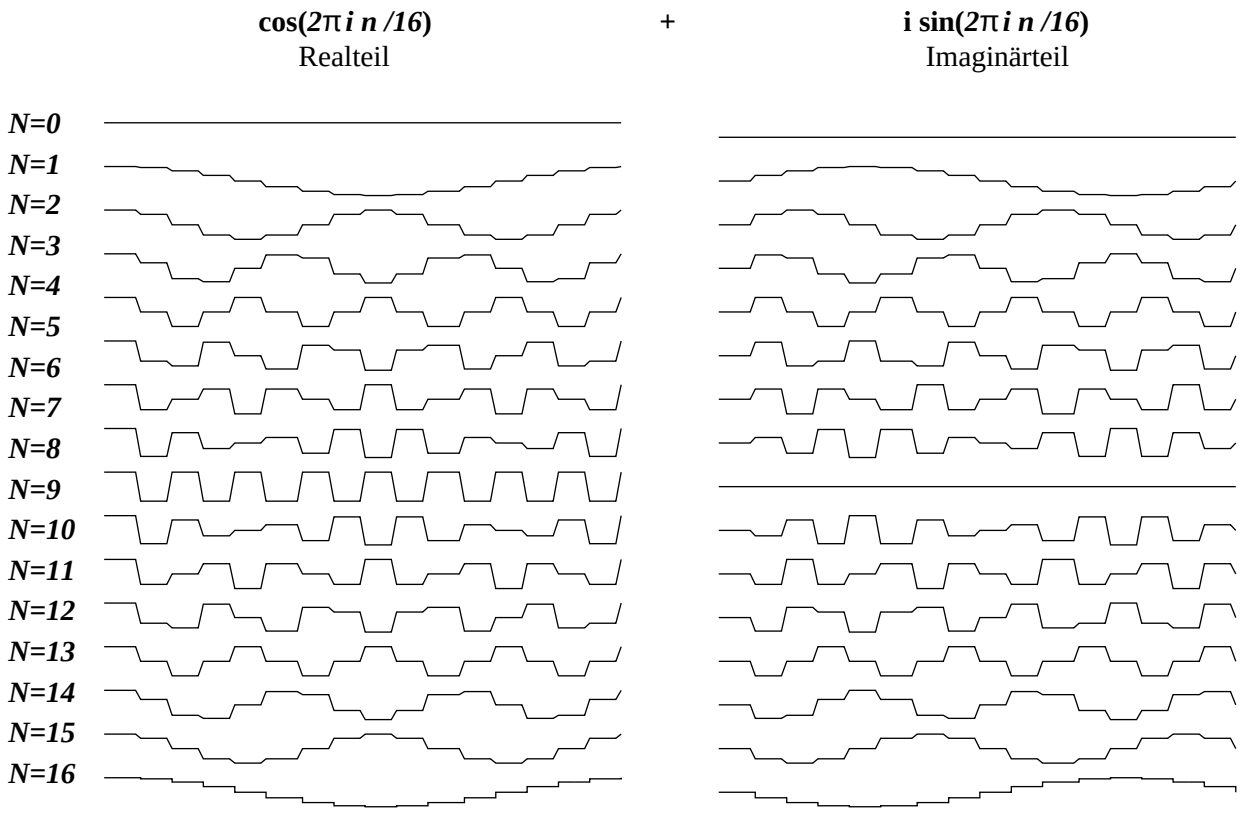
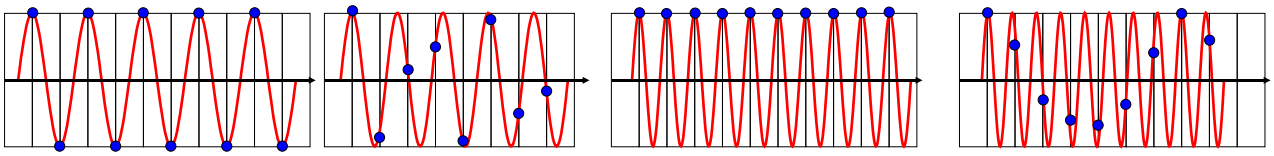


Abb. 3.5: alle Basisfunktionen in Real- und Imaginärteil für eine $N = 16$ DFT

Man kann sich leicht überlegen, welche Wellenzahlen k überhaupt erlaubt sein können. Dazu muss man betrachten, welche Schwingungen noch auf einem Gitter mit N Stützpunkten darstellbar sind. Die schnellste Schwingung ist sicher diejenige, bei der genau zwei Stützstellen auf einer Schwingung liegen. Liegen weniger Stützstellen auf einer (dann noch schnelleren) Schwingung, so kann diese Frequenz nicht dargestellt werden (siehe Abb. 3.6).



gerade noch darstellbar (exakt 2 Stützstellen) | etwas zu hoch (< 2 Stützstellen) | doppelte Grenzfrequenz (1 Stützstellen) | noch etwas höher (~0,8 Stützstellen)

Abb. 3.6: Mit der DFT können nur periodische Strukturen bis zu einer gewissen Grenzfrequenz (2 Bildpunkte pro Schwingung) dargestellt werden.

Die Bilder in Abb. 3.6 zeigen, dass es nicht möglich ist sehr hohe Frequenzen mit N Stützstellen darzustellen. Bei sehr hohen Frequenzen können sogar zu niedrige Frequenzen in der Darstellung erzeugt werden (Schwebungen, Abtasttheorem). Die niedrigste darstellbare Schwingung ist diejenige, bei der exakt eine Wellenlänge auf der Breite des Bildes unterkommt. Ein Beispiel ist der Realteil der Basisfunktion zu $N = 1$.

Bemerkungen zu Datentypen/dynamischer Bereich: Man geht von einem Grauwertbild mit 256 verschiedenen Grauwerten aus. Wird dieses transformiert, so ergibt sich ein komplexes Bild (Fourier-Transformierte). Dieses Bild kann zwar wieder mit 256 Abstufungen dargestellt werden, dies ist aber

nicht empfehlenswert, da die Werte sehr stark streuen (vergleichsweise großer dynamischer Bereich), sodass in dem meisten Fällen eine Darstellung als `float`-Bild empfehlenswert ist.

3.5 Mehrdimensionale Fourier-Transformationen

Die Fourier-Trafo wurde bisher nur für eine Dimension betrachtet. Die Erweiterung auf mehrere Dimensionen (speziell zwei für übliche Bilder) ist einfach möglich: Man transformiert zunächst in der x- und danach in der y-Richtung. Man erhält so:

Satz 3.4 (W-dimensionale Fourier-Transformation) *Sei $g : \mathbb{R}^W \to \mathbb{C}$ eine quadratintegrable Funktion, also $\int\limits_{-\infty}^{\infty} |g(\vec{x})|^2 \, \mathrm{d}^W x < \infty$. Die Integration bedeutet hier eine Integration über das gesamte, unendliche Volumen des $\mathbb{R}^W$. Man kann also auch schreiben: $\int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} \ldots |g(\vec{x})|^2 \, \mathrm{d}x \, \mathrm{d}y \ldots < \infty$. Dann gilt für die Fourier-Transformierte $\hat{g}(\vec{k})$:*

$$\hat{g}(\vec{k}) = \mathcal{F}[g] = \int\limits_{-\infty}^{\infty} g(\vec{x}) \cdot \mathrm{e}^{-2\pi \mathrm{i} \vec{k} \cdot \vec{x}} \, \mathrm{d}^W x =: \langle g(\vec{x}), \mathrm{e}^{2\pi \mathrm{i} \vec{k} \cdot \vec{x}} \rangle$$

Die Rücktransformation ergibt analog:

$$g(\vec{x}) = \mathcal{F}^{-1}[\hat{g}] = \int\limits_{-\infty}^{\infty} \hat{g}(\vec{k}) \cdot \mathrm{e}^{2\pi \mathrm{i} \vec{k} \cdot \vec{x}} \, \mathrm{d}^W x$$

Die einzige Veränderung gegenüber der eindimensionalen Transformation ist also, dass das Produkt kx im Exponenten durch ein Skalarprodukt der Größen $\vec{x}$ und $\vec{k}$ ersetzt wurde. Die Bedeutung von $\vec{x}$ ist klar. Es ist einfach nur der Ortsvektor, der auf einen bestimmte Position in $g(\cdot)$ zeigt. Der Vektor $\vec{k}$ wird **Wellenvektor**, siehe dazu Abschnitt 3.1.2.

Man erhält aus obigen Satz auch die Form der zweidimensionalen, diskreten Fourier-Trafo. Die Erweiterung auf $W > 2$ Dimensionen ist vollkommen natürlich möglich. Der Fall $W = 2$ ist aber sehr wichtig und wird deswegen hier explizit angegeben. Die Angabe der höheren Dimensionen ergibt im wesentlichen nur Schwierigkeiten in der Notation.

Satz 3.5 (2-dimensionale diskrete Fourier-Transformation (2D-DFT)) *Man geht von einem zweidimensionalen Bild $g_{m,n}$ aus, das sich als $M \times N$-Matrix darstellt. Die 2D-DFT bildet dann auf folgendes Bild (Fourier-Transformierte) ab:*

$$\hat{g}_{u,v} = \frac{1}{MN} \sum_{n=0}^{N-1} \left[\sum_{m=0}^{M-1} g_{m,n} \cdot \exp\left(-\frac{2\pi \mathrm{i} m u}{N}\right) \right] \cdot \exp\left(-\frac{2\pi \mathrm{i} n v}{N}\right) \qquad 0 \le u < M, \;\; 0 \le v < N$$

Die Rück-Transformation ist analog:

$$g_{m,n} = \sum_{v=0}^{N-1} \left[\sum_{u=0}^{M-1} \hat{g}_{u,v} \cdot \exp\left(\frac{2\pi \mathrm{i} m u}{N}\right) \right] \cdot \exp\left(\frac{2\pi \mathrm{i} n v}{N}\right) \qquad 0 \le m < M, \;\; 0 \le n < N$$

Durch die Klammerung tritt die Struktur der Hintereinanderausführung klar zu Tage. Um auch dies wieder als Skalarprodukt zu schreiben muss man ein Skalarprodukt auf dem Raum der 2D-Matritzen

definieren. Es lautet:

$$\langle g, f \rangle = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} g^*_{m,n} f_{m,n}$$

Damit gilt dann:

$$\hat{g}_{u,v} = \frac{1}{\sqrt{MN}} \cdot \langle b_{u,v}, g \rangle \quad \text{und} \quad g_{m,n} = \frac{1}{\sqrt{MN}} \cdot \langle b_{-m,-n}, \hat{g}_{u,v} \rangle$$

Die Matrix der Basismatritzen ergibt sich als:

$$b_{u,v} = \exp\left(\frac{2\pi \mathrm{i} v}{N}\right) \cdot \exp\left(\frac{2\pi \mathrm{i} v}{M}\right)$$

Die folgende Abb. 3.7 zeigt Real- und Imaginärteil der Basismatritzen für eine 16×16-2D-DFT, also die Basisbilder:

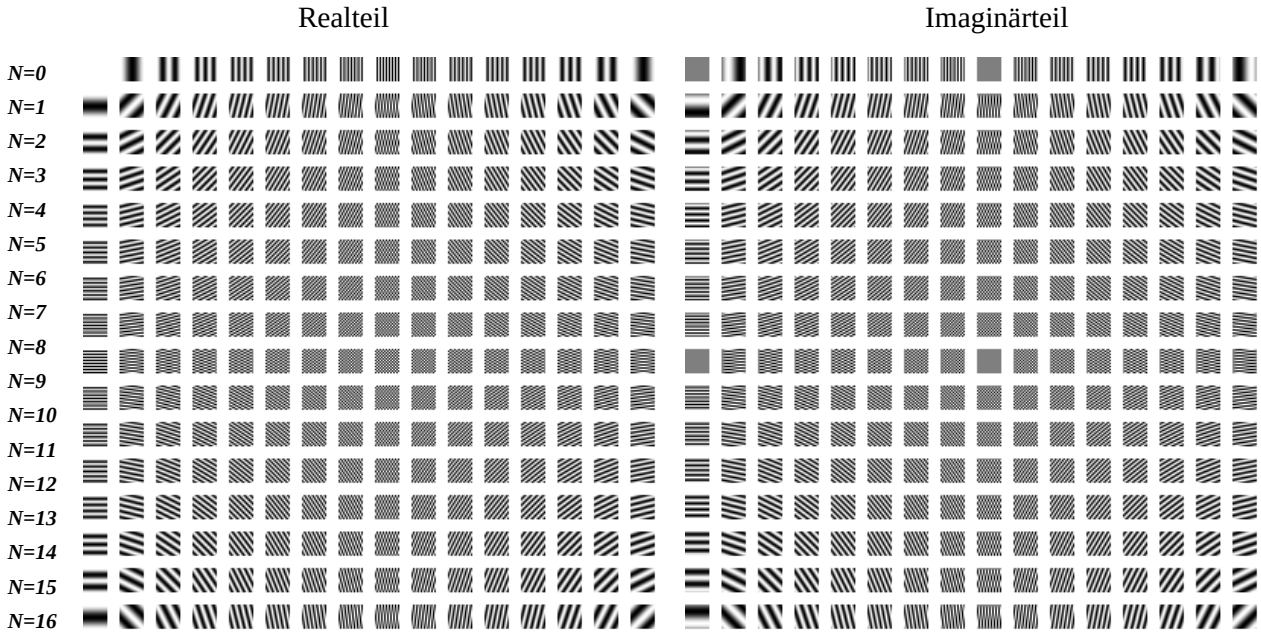


Abb. 3.7: Real- und Imaginärteil der Basismatritzen für eine 16×16-2D-DFT

Man kann beobachten, dass Basisbilder, die den gleichen Abstand zur linken oberen Ecke haben, in etwa gleiche räumliche Frequenz haben. Nur ihr Verkippungswinkel variiert. Je näher das Bild zur Mitte ist, desto höher ist die Frequenz. Die erste Zeile/Spalte ergibt bei einem Schnitt durch das Bild die Basisfunktionen der 1D-DFT aus dem vorherigen Abschnitt. Diese zwei Darstellungen sind also analog.

3.6 Eigenschaften der Fourier-Transformation

3.6.1 Allgemeine Eigenschaften

Betrag und Phase im Fourierraum: Auch wenn die Funktion $g(x)$ eine rein reelle Funktion ist, kann ihre Fourier-Transformierte $\hat{f}(k)$ durchaus einen nicht-verschwindenden Imaginärteil haben. Die gesamte Bildinformation steckt im Realraum also im Realteil. Im Fourierraum verteilt sich diese Information auf Real- $\hat{a}(k) = \mathrm{Re}\ \hat{g}(k)$ und Imaginärteil $\hat{b}(k) = \mathrm{Im}\ \hat{g}(k)$, bzw. Phase $\phi = \arg(\hat{g}(k)) = \arctan \hat{b}(k)\hat{a}(k)$ und Betrag $|\hat{g}(k)|$ der Fourier-Transformierten $\hat{g}(k) = \hat{a}(k) + \mathrm{i}\hat{b}(k) = |\hat{g}(k)| \cdot \mathrm{e}^{\mathrm{i} \arg(\hat{g}(k))}$. Es darf also auf keinen Fall einer der beiden Anteile vernachlässigt werden. Die folgende AAbb. 3.8 zeigt,

was passiert, wenn man zunächst $\hat{g}$ berechnet und dann vor der Rücktransformation den Real- oder den Imaginärteil 0 setzt, also vernachlässigt.

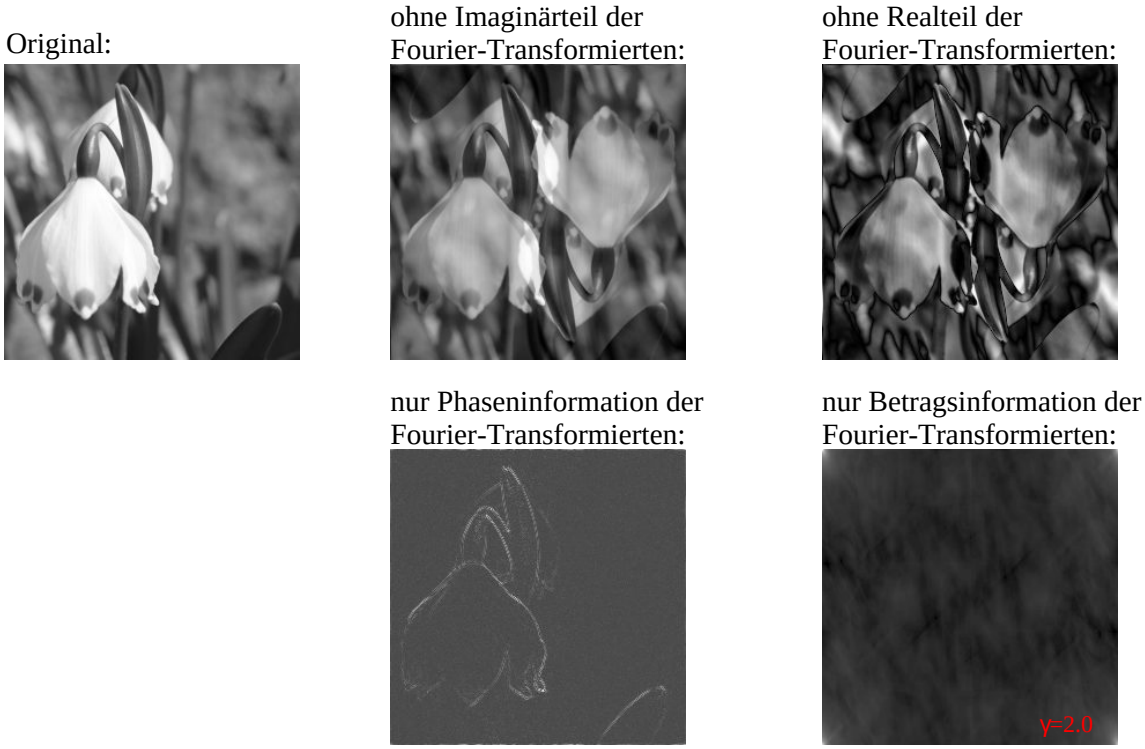


Abb. 3.8: Auswirkung der vernachlässigung von Real- und Imaginäreteil, bzw. Betrag oder Phase im Fourierraum.

Diese Bilder bestätigen nochmals, dass sich die Information im Foruierraum zu gleichen Teilen auf Real- und Imaginärteil verteilt.

Symmetrien: Die Fouriertransformation weist einige Symmetrien auf, die auch von praktischen Nutzen sein können, da man evtl. einige Daten weglassen kann. Es treten zwei Gruppen von Symmetrien auf:

gerade	$g(-\vec{x}) = g(\vec{x})$	**ungerade**	$g(-\vec{x}) = -g(\vec{x})$
hermitesch	$g(-\vec{x}) = g^*(\vec{x})$	**antihermitesch**	$g(-\vec{x}) = -g^*(\vec{x})$

Damit ergeben sich folgende Symmetriebeziehungen für die Fourier-Transformation:

reell	∘–•	hermitesch	hermitesch	∘–•	reell
imaginär	∘–•	antihermitesch	antihermitesch	∘–•	imaginär
gerade	∘–•	gerade	ungerade	∘–•	ungerade
reell und gerade	∘–•	reell und gerade	reell und ungerade	∘–•	imaginär und ungerade
imaginär und gerade	∘–•	imaginär und gerade	imaginär und ungerade	∘–•	reell und gerade

Diese können so ausgenutzt werden, dass es etwa für ein reelles Bild nicht nötig ist den gesamten Fourier-Raum zu speichern. Es genügt eine Hälfte, da die restlichen Pixel berechnet werden können. Dies ist auch einsichtig, da man im Fourierraum zu jedem Pixel zwei Werte (Real- und Imaginärteil) speichert, während im Realraum nur ein Wert nötig war. Man würde sonst mit der FT unnötige Information generieren.

Die Auswirkungen der obigen Symmetrien kann man z.B. auch in Abb. 3.7 sehen. Lässt man den Imaginärteil im Fourierraum weg, so erzeugt man dort ein rein reales Bild, das nach der Rücktransformation eine Symmetrie aufweist.

Separabilität:

Satz 3.6 (Separabilität der Fourier-Transformation) *Als separable Funktion bezeichnet man eine Funktion mit folgender Eigenschaft:*

$$f(x,y) = g_1(x) \cdot g_2(y).$$

Für die Fourier-Transformation einer solchen Funktion gilt:

$$\boxed{f(x,y) = g_1(x) \cdot g_2(y) \quad \circ\!\!-\!\!\bullet \quad \hat{f}(k_x,k_y) = \hat{g}_1(k_x) \cdot \hat{g}_2(k_y)}$$

Dies lässt sich leicht zeigen. Man benutzt dazu die Separabilität des Kerns der Fouriertransformation:

$$\begin{aligned}
\hat{f}(k_x,k_y) = \mathcal{F}[f(x,y)] &= \iint f(x,y) \cdot \mathrm{e}^{2\pi\mathrm{i}(k_x x + k_y y)} \ \mathrm{d}y \ \mathrm{d}x = \\
&= \iint g_1(x) \cdot g_2(y) \cdot \mathrm{e}^{2\pi\mathrm{i}k_x x} \cdot \mathrm{e}^{2\pi\mathrm{i}k_y y} \ \mathrm{d}y \ \mathrm{d}x = \\
&= \left\{ \int g_1(x) \cdot \mathrm{e}^{2\pi\mathrm{i}k_x x} \ \mathrm{d}x \right\} \cdot \left\{ \int g_2(y) \cdot \mathrm{e}^{2\pi\mathrm{i}k_y y} \ \mathrm{d}y \right\} = \\
&= \mathcal{F}[g_1(x)] \cdot \mathcal{F}[g_2(y)] = \hat{g}_1(k_x) \cdot \hat{g}_2(k_y)
\end{aligned}$$

Verschiebung:

Satz 3.7 (Verschiebungstheorem) *Wird eine Funktion $g(\vec{x})$ um einen Vektor $\vec{x}_0$ verschoben, so erhält man die neue Funktion $h(\vec{x}) = g(\vec{x} - \vec{x}_0)$. Für die Fourier-Transformierte gilt dann:*

$$\boxed{\mathcal{F}[h(\vec{x})] = \mathcal{F}[g(\vec{x} - \vec{x}_0)] \quad \circ\!\!-\!\!\bullet \quad \hat{g}(\vec{k}) \cdot \mathrm{e}^{-2\pi\mathrm{i}\vec{x}_0 \cdot \vec{k}} = \hat{h}(\vec{k})}$$

Das bedeutet, dass die Fourier-Transformierte einer Funktion, durch die Verschiebung der Funktion nur eine zusätzliche Phase bekommt, die von der Verschiebung abhängt. Umgekehrt bedeutet eine Verschiebung der Fourier-Transformierten um $\vec{k}_0$ eine Modulation des Bildes im Ortsraum mit einer Schwingung der Wellenzahl $\vec{k}_0$.

Diese Eigenschaft lässt sich ebenfalls recht leicht beweisen, wenn man eine Variablentransformation $\vec{u} = \vec{x} - \vec{x}_0$ durchführt:

$$\begin{aligned}
\mathcal{F}[g(\vec{x} - \vec{x}_0)] &= \iint g(\vec{x} - \vec{x}_0) \cdot \mathrm{e}^{-2\pi\mathrm{i}\vec{k}\cdot\vec{x})} \ \mathrm{d}^W x = \\
&= \iint g(\vec{u}) \cdot \mathrm{e}^{-2\pi\mathrm{i}\vec{k}\cdot(\vec{u}+\vec{x}_0)} \ \mathrm{d}^W u = \\
&= \mathrm{e}^{-2\pi\mathrm{i}\vec{k}\cdot\vec{x}_0} \iint g(\vec{u}) \cdot \mathrm{e}^{2\pi\mathrm{i}\vec{k}\cdot\vec{u}} \ \mathrm{d}^W u = \mathrm{e}^{-2\pi\mathrm{i}\vec{k}\cdot\vec{x}_0} \cdot \hat{g}(\vec{k})
\end{aligned}$$

Aufteilung in Sinus- und Cosinus-Transformation: Jede beliebige Funktion $g(\vec{x})$ lässt sich in einen geraden Anteil $g_g(\vec{x})$ und einen ungeraden Anteil $g_u(\vec{x})$ aufteilen. Diese berechnen sich so:

$$g_g(\vec{x}) = \frac{g(\vec{x}) + g(-\vec{x})}{2} \quad \text{und} \quad g_u(\vec{x}) = \frac{g(\vec{x}) - g(-\vec{x})}{2}$$

Damit kann man die Fourier-Transformation in eine Cosinus- und eine Sinustransformation zerlegen:

$$\boxed{hatg(\vec{k}) = 2\int_0^\infty g_g(\vec{x})\cos(2\pi\cdot\vec{k}\cdot\vec{x})\,\mathrm{d}^W x + 2\mathrm{i}\int_0^\infty g_u(\vec{x})\sin(2\pi\cdot\vec{k}\cdot\vec{x})\,\mathrm{d}^W x}$$

Hieraus kann man ersehen, dass für die Darstellung einer ungeraden Funktion der $\sin$-Anteil von $e^{\mathrm{i}\varphi} = \cos\varphi + \mathrm{i}\cdot\sin\varphi$ ausreicht (der Sinus ist selber ungerade) und für eine gerade Funktion der $\cos$-Anteil.

Faltungstheorem:

Satz 3.8 (Faltungstheorem) *Für die Bildverarbeitung wird das **Faltungstheorem** von Bedeutung sein. Es besagt, dass sich eine Faltung $f(x) \circledast h(x)$ im Fourier-Raum als Multiplikation darstellt:*

$$\boxed{\mathcal{F}[f(x)\circledast h(x)] = \hat{f}(k)\cdot\hat{h}(k) \quad \Leftrightarrow \quad f(x)\circledast h(x) \circ\!\!-\!\!\bullet\ \hat{f}(k)\cdot\hat{h}(k)}$$

Ableitung:

Satz 3.9 (Fourier-Transformation der Ableitung) *Die Fourier-Transformierte der Ableitung $\frac{\partial g}{\partial x}$ einer Funktion $g(x)$ hat eine sehr einfache Form, wenn die Fourier-Transformierte $\hat{g}(k)$ von $g(x)$ bekannt ist:*

$$\frac{\partial g}{\partial x} \quad \circ\!\!-\!\!\bullet \quad 2\pi\mathrm{i}\cdot k\cdot\hat{g}(k)$$

Eine Multiplikation im Fourier-Raum entspricht einer Faltung im Real-Raum, sodass man weiter erhält:

$$\frac{\partial g}{\partial x} = g(x)\circledast\mathcal{F}^{-1}[2\pi\mathrm{i}\cdot k]$$

Dieses Ergebnis kann zur Erzeugung von Ableitungsfiltern angewendet werden.

Glattheit und Kompaktheit: Je glatter eine Funktion ist, desto weniger abrupte Sprünge und schnelle Anstiege enthält sie. In Abb. 3.9 ist dies veranschaulicht. Für einen schnellen Anstieg in einer Funktion $g(x)$ benötigt man Anteile mit hoher Frequenz in ihrer Fourier-Transformierten $\hat{g}(x)$. Damit ist die Fourier-Transformierte einer solchen Funktion sehr breit. Je glatter die Funktion $g(x)$ ist, desto weniger hohe Frequenzanteile werden benötigt und die Fourier-Transformierte wird schmaler (kompakter).

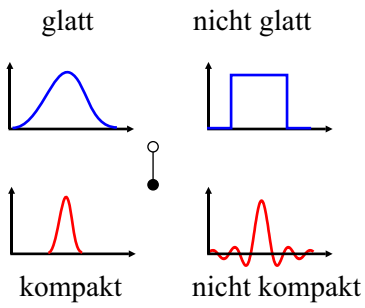


Abb. 3.9: glatte und weniger glatte Funktion (oben) und ihre kompakte bzw. ausgedehnte Fourier-Transformierte

3.6.2 Spezielle Eigenschaften der diskreten Transformationen

Symmetrie des Kerns: Die Funktion $\mathrm{e}^{-2\pi \mathrm{i} n/N}$ wird als Kern der DFT bezeichnet. Dieser kern weist eine charakteristische Periodizität auf:

$$\mathrm{e}^{-2\pi \mathrm{i} n/N} = \mathrm{e}^{-2\pi \mathrm{i}(n+lN)/N}, \quad \text{mit} \quad l \in \mathbb{N}$$

Dies bedeutet, dass sich das Bild der Fourier-Transformierten im Wellenzahlraum periodisch wiederholt. Dies lässt sich für die 1D-DFT graphisch als Ring darstellen (siehe Abb. 3.10).

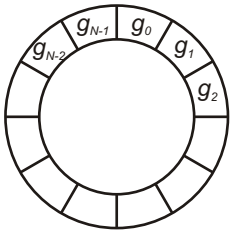


Abb. 3.10: geometrische Interpretation der Periodizität des Kerns der 1D-DFT

3.7 DFT als diskrete unitäre Transformation und weitere Transformationen

Die Fourier-Transformation (FT) ist ein Vertreter einer großen Klasse von Transformationen, den sog. **unitären Transformationen**. Sie entsprechen in niedrig-dimensionalen Räumen den Rotationen. Es handelt sich also um Längen- und Winkel-erhaltende Transformation. Diese unitären Transformationen auf endlich dimensionalen (diskreten) Vektorräumen können folgendermaßen definiert werden:

Definition 3.1 (unitäre Transformation) *Eine unitäre Transformation (durch eine Matrix $\mathbf{U}$ dargestellt) ist eine lineare bijektive Abbildung von einem endlich-dimensionalen Vektorraum V auf sich selbst. Es gilt:*

- $\mathbf{U}$ *ist eine unitäre Matrix* ($\mathbf{U}^{-1} = \mathbf{U}^t$)
- *die Transformation erhält das Skalarprodukt auf V: $\langle \vec{g}, \vec{h} \rangle = \langle \mathbf{U}\vec{g}, \mathbf{U}\vec{h} \rangle$ für alle $\vec{g}, \vec{h} \in V$. Dies bedeutet die Erhaltung des Winkels zwischen $\vec{g}$ und $\vec{h}$.*
- *Aus dem letzten Punkt ergibt sich auch die Erhaltung der Länge eines Vektors: $\|\vec{g}\|_2 = \sqrt{\langle \vec{g}, \vec{g} \rangle} = \sqrt{\langle \mathbf{U}\vec{g}, \mathbf{U}\vec{g} \rangle} = \|\mathbf{U}\vec{g}\|_2$*
- *Die Komposition zweier Transformationen $\mathbf{U}_1 \cdot \mathbf{U}_2$ ist ebenfalls eine unitäre Transformation.*
- *die Zeilen- und Spaltenvektoren von $\mathbf{U}$ bilden eine Orthonormalbasis von V.*

Man kann nun die DFT etwas umformulieren und sie somit auf die Form einer unitären Transformation darstellen (mit $w_N = \mathrm{e}^{2\pi \mathrm{i}/N}$):

$$\begin{aligned} \hat{g}_k &= \frac{1}{N} \sum_{n=0}^{N-1} g_n \cdot \mathrm{e}^{-2\pi \mathrm{i} nk/N} = \\ &= \frac{1}{N} \sum_{n=0}^{N-1} g_n \cdot w_N^{-nk} \end{aligned}$$

Dies kann man als Matrixmultiplikation darstellen:

$$\hat{\vec{g}} = \frac{1}{N} \cdot \mathbf{W}_N \vec{g}$$

Dabei ist

$$\mathbf{W}_N = \begin{pmatrix} w_N^{-0\cdot 0} & w_N^{-0\cdot 1} & \cdots & w_N^{-0\cdot N} \\ w_N^{-1\cdot 0} & w_N^{-1\cdot 1} & \cdots & w_N^{-1\cdot N} \\ \vdots & \vdots & \ddots & \vdots \\ w_N^{-N\cdot 0} & w_N^{-N\cdot 1} & \cdots & w_N^{-N\cdot N} \end{pmatrix} = \begin{pmatrix} w_N^0 & w_N^0 & \cdots & w_N^0 \\ w_N^0 & w_N^{-1} & \cdots & w_N^{-N} \\ \vdots & \vdots & \ddots & \vdots \\ w_N^0 & w_N^{-N} & \cdots & w_N^{-N^2} \end{pmatrix}$$

Nutzt man zusätzlich noch die Periodizität des Kerns der DFT aus (siehe 3.6), so kann man $\mathbf{W}_N$ weiter vereinfachen (hier am Beispiel $N = 4$):

$$\mathbf{W}_4 = \begin{pmatrix} w_4^{-0} & w_4^{-0\cdot 1} & w_4^{-0\cdot 2} & w_4^{-0\cdot 3} \\ w_4^{-0\cdot 1} & w_4^{-1\cdot 1} & w_4^{-2\cdot 1} & w_4^{-3\cdot 1} \\ w_4^{-0\cdot 2} & w_4^{-1\cdot 2} & w_4^{-2\cdot 2} & w_4^{-3\cdot 2} \\ w_4^{-0\cdot 3} & w_4^{-1\cdot 3} & w_4^{-2\cdot 3} & w_4^{-3\cdot 3} \end{pmatrix} = \begin{pmatrix} w_4^0 & w_4^0 & w_4^0 & w_4^0 \\ w_4^0 & w_4^{-1} & w_4^{-2} & w_4^{-3} \\ w_4^0 & w_4^{-2} & w_4^{-4} & w_4^{-6} \\ w_4^0 & w_4^{-3} & w_4^{-6} & w_4^{-9} \end{pmatrix} = \begin{pmatrix} w_4^0 & w_4^0 & w_4^0 & w_4^0 \\ w_4^0 & w_4^3 & w_4^2 & w_4^1 \\ w_4^0 & w_4^2 & w_4^0 & w_4^2 \\ w_4^0 & w_4^1 & w_4^2 & w_4^3 \end{pmatrix}$$

Damit reduziert sich die Berechnung einer diskreten Fourier-Transformierten $\hat{\vec{g}}$ aus einem Datenvektor $\vec{g}$ entspricht also einer Matrixmultiplikation mit einer fest vorgegebenen Matrix $\mathbf{W}_N$. Diese lässt sich mit dem Aufwand $\mathcal{O}(N^2)$ berechnen. Nutzt man zusätzlich bekannte Eigenschaften dieser speziellen Basis, so kommt man zu schnellen Algorithmen, die als **Fast-Fourier-Transformationen (FFT)** bezeichnet werden. Sie arbeiten mit der Komplexität $\mathcal{O}(N \cdot \log N)$.

Es gibt noch weitere Basis-Systeme, die durch eine unitäre Transformation erreichbar sind. Dazu zählen:

- Sinus- und Cosinus-Transformation
- Hartley-Transformation
- Hadamardtransformation
- Haartransformation

3.8 Lokale Fourier-Transformation

Die Fourier-Transformation führt ein Bild in Ortsdarstellung in ein Bild in Frequenzdarstellung über. Dabei ist die Orts-Information im Ortsraum lokal, während die Frequenzinformation über den ganzen Ortsraum verschmiert ist. Im Fourierraum gilt das umgekehrt. Manchmal möchte man aber eine gemischt Darstellung erreichen, in der sowohl lokale Orts- als auch Frequenzinformation vorliegt. Dazu führt man die lokale Fourier-Transformation ein:

$$\hat{g}(\vec{x}, \vec{k}_0) = \int_{-\infty}^{\infty} g(\vec{x}')w(\vec{x} - \vec{x}')\, \mathrm{e}^{-2\pi \mathrm{i} \vec{k}_0 \cdot \vec{x}'} \; \mathrm{d}x'^2$$

Dabei ist $w(\vec{x} - \vec{x}')$ eine Fensterfunktion, die nur in einer kleinen Umgebung von $\vec{x}'$ von 0 verschieden ist. Die Fourier-Transformation wird so auf die lokale Nachbarschaft eingeschränkt. Es lässt sich zeigen, dass dies einer Bandpass-gefilterten Darstellung des Bildes entspricht, weil nur Wellenzahlen berücksichtigt werden, die innerhalb des $\neq 0$-Bereiches des Fensters Platz finden. Die Breite des Bandpasses ist umgekehrt propotional zur Breite des Fensters.

4 Digitalisierung, Abtastung und Quantisierung

Bisher wurden kontinuierliche und diskrete Daten unterschieden. In der praktischen Anwendung der Bildverarbeitung tauchen aber eigentlich nur diskrete Daten auf, weil Computer i.A. nur mit diskreten Werten rechnen. Gleitkommazahlen sind zwar quasi-kontinuierlich, aber auch hier gibt es kleinste diskrete Schritte. Außerdem liefern übliche Signalquellen, wie etwa Kameras oder Digital-/Analog-Wandler (z.B. als PCI-Einsteckkarten, oder auf der Soundkarte) diskrete Werte, die meist mit konstantem Abstand und einer bestimmten Länge geliefert werden: So geben übliche CCD-Kameras Bilder mit 8-16 Bit aus, das entspricht 256-65536 Graustufen, deren Abstände in erster Näherung konstant sind. Es stellt sich nun die Frage, wie man diese Umwandlung der realen (kontinuierlichen) Daten in diskrete Daten auf dem Computer verstehen kann. Dieser Umwandlungsprozess gliedert sich in mehrere Schritte:

1. **(Bilderzeugung):** Dieser Teil beschreibt, wie das Messsystem (Kamera) die realen Daten (z.B. mittels eines Objektives) auf einen Sensor (CCD-Chip in der Kamera) abbildet.
2. **Abtastung/Digitalisierung:** Hier wird das reale Bild $g(\vec{x})$ an verschiedenen, diskreten Punkten vermessen. Eine Kamera hat z.B. 1024×1024 Pixel an denen sie die Intensität misst. Ein weitere Beispiel ist eine Soundkarte, die ein Sprachsignal nur in bestimmten Zeitintervallen abtastet. Es ergibt sich also eine erste Form der Diskretisierung, nämlich auf einem Gitter. Wie das obige Beispiel der Kamera zeigt, wird auch nicht der gesamte Raum abgetastet, sondern es wird evtl. nur ein Fenster ausgeschnitten (z.B. Größe des CCD-Sensors).
3. **Quantisierung:** Die Werte an den Gitterpunkten sind in diesem Modell noch kontinuierlich. Sie müssen nun noch in digitale Werte (z.B. 1 Byte $[0..255]$) überführt werden.

4.1 Bilderzeugung, Abtastung

Zur Bilderzeugung zählen zwei Schritte: Zunächst wird das zu vermessende Objekt auf einen Sensor abgebildet. Bei einer Kamera übernimmt das z.B. das Objektiv. Hierbei können Verzerrungen und andere Einflüsse auf das (noch kontinuierliche) Bild auftreten, die hier aber nicht weiter besprochen werden sollen, da hierzu ein genaues Verständnis z.B. der Abbildungseigenschaften von Linsen nötig wäre. Danach wird das Bild auf den Sensor projiziert. Das Bild auf dem Sensor wird mit $g_{\text{CCD}}(\vec{x})$ bezeichnet. Hier soll der Fall einer Kamera besprochen werden. Die folgende Abb. 4.1 verdeutlicht den Aufbau einer CCD-Kamera und wie damit abgetastet wird.

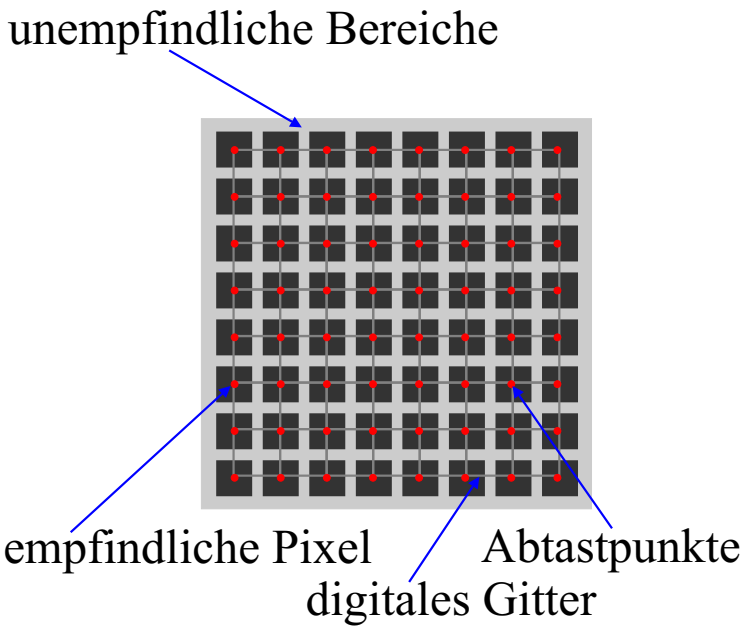


Abb. 4.1: Abtastung mit einer CCD-Kamera

Eine moderne Digitalkamera ist als Gitter von lichtempfindlichen Zellen aufgebaut (dunkelgrau). Zwischen den lichtempfindlichen Bereichen befinden sich unempfindliche Stege (hellgrau). Pro Pixel wird nur ein Wert abgelesen, der dem Abtastpunkt (rot) zugeordnet wird. Dies bedeutet eine Mittelung der Intensität des einfallenden Lichtes über die Fläche des Pixels. Da nur die Werte an den Abtastpunkten zurückgeliefert werden, entsteht ein Gitter von Werten (digitales Gitter, hellgrau). Die Mittelung über einen einzelnen Pixel kann mathematisch als Faltung mit einem Rechteck $\text{rect}(\vec{x}) = \begin{cases} 1 & \vec{x} \text{ im Pixel} \\ 0 & \text{sonst} \end{cases}$ dargestellt werden:

$$g(\vec{x}_{m,n}) = \text{rect} \circledast g_{\text{CCD}} = \int_{-\infty}^{\infty} \text{rect}(\vec{x}' - \vec{x}_{n,m}) \cdot g_{\text{CCD}}(\vec{x}') \, \mathrm{d}^P x'$$

Dabei ist $\vec{x}_{m,n} = \begin{pmatrix} m \cdot \Delta x \\ n \cdot \Delta y \end{pmatrix}$, $m, n \in \mathbb{Z}$ ein Punkt auf dem Diskretisierungsgitter und $\text{rect}(\vec{x}' - \vec{x}_{n,m})$ ist 1 nur für $\vec{x}'$ in dem ihm umgebenden Pixel. Die Integration erfolgt in P Dimensionen über den gesamten Raum. Für eine Kamera ist $P = 2$.

Nach diesem Schritt liegt also ein Bild $g_{m,n} \equiv g(\vec{x}_{m,n})$ vor, wobei die Werte an den Positionen n, m des Gitters noch kontinuierlich sind. Bei der Abtastung kann vor Allem ein Problem auftauchen, das vom folgenden **Abtasttheorem** beschrieben wird:

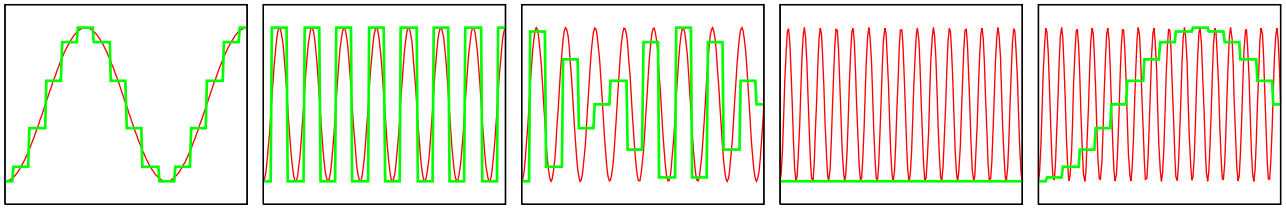
Satz 4.1 (Abtasttheorem) *Ist das Spektrum (Fourier-Transformierte) $\hat{g}(\vec{k})$ eines Signales $g(\vec{x})$ bandbegrenzt, d.h.*

$$\hat{g}(\vec{k}) = 0 \quad \forall \left|\vec{k}\right| \geq \frac{k_{max}}{2} = \frac{1}{\lambda_{min} \cdot 2},$$

dann kann es aus einer Abtastung mit der Schrittweite von $\Delta x = \frac{1}{k_{max}}$ exakt rekonstruiert werden.

Dieser Satz besagt, dass man nur ein solches Signal $g(\vec{x})$ aus einer Abtastung richtig rekonstruieren kann, dessen kleinste periodische Strukturen (charakterisiert durch die minimale Wellenlänge λ_{min}, bzw. die maximale Wellenzahl/räumliche Frequenz $k_{\text{max}} = \frac{1}{\lambda_{\text{min}}}$) immernoch zwei Abtastpunkte enthalten. Die Bandbegrenztheit eines solchen Signals besagt, dass keine Strukturen vorkommen, die eine größere Frequenz enthalten, als die Grenzfrequenz. Dies kann in der obigen einfachen Form ausgedrückt werden, die das Fourier-transformierte Signal $\hat{g}(\vec{k})$ benutzt, das angibt, wie stark eine Frequenzkomponente $\vec{k}$ im Signal vorhanden ist.

Die folgende Abb. 4.2 zeigt anschaulich einige Fehler, die beim Abtasten mit zu groben 1D-Gittern auftreten können:



< Grenzfrequenz | Grenzfrequenz | etwas zu hoch | doppelte Grenzfrequenz | noch etwas höher

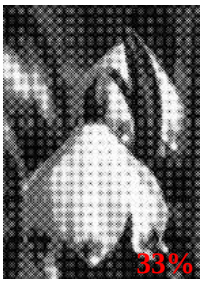
Abb. 4.2: Fehler bei der 1D-Abtastung (Aliasing-Effekte)

Die Grenzfrequenz kann gerade noch richtig dargestellt werden. Bei etwas höheren Frequenzen treten Sturkturen auf, die vorher nicht da waren. Die doppelte Grenzfrequenz wird nicht gar nicht erkannt und bei noch höheren Frequenzen sieht man im Bild Strukturen sehr viel größerer Wellenlänge. Letzteres rührt daher, dass in jeder Schwingungsperiode nur noch etwas weniger als ein Abtastpunkt liegt, der immer weiter innerhalb der Schwingung wandert. Solche Fehler werden (im 1-dimensionalen) als **Aliasing** bezeichnet.

Auch in höher-dimensionalen Signalen können solche Fehler auftreten. Man spricht dann oft von **Moiré-Musternn**. Sie treten z.B. auf, wenn man ein Bild aus einer Zeitschrift (Rasterdruck) mit nur geringfügig größerer Auflösung einscannt. Man überlagert dann das Druck-Gitter mit dem Abtastgitter des Scanners. Solche Erscheinungen treten immer auf, wenn man zwei Gitter überlagert. Die folgende Abb. 4.3 zeigt Beispiele für den Moiré-Effekt

Moiré-Muster an einem Vorhang:

Moiré-Effekt beim Rasterdruckes:



verschiedene Zoomstufen/Abtastintervalle

Abb. 4.3: Falte eines Vorhangs (dünnes Gitter des Stoffes), mit Moiré-Mustern (mit freundlicher Genehmigung von Julia Ziegler) und Moiré-Effekte beim Rasterdruck

Die Bilderzeugung lässt sich als Faltung mit einer Rechteck-Funktion sehen. Die Abtastung ist analog ein Multiplikation mit einem sog. Delta-Kamm $\sum_{m,n} \delta(\vec{x} - \vec{r}_{m,n})$ (dabei sind $\vec{r}_{m,n}$ die Abtastpunkte), oder auch „Nagelbrettfunktion“. Das ist eine Funktion, die sich aus δ-Peaks an den Abtastpunkten $\vec{r}_{m,n}$ zusammensetzt. Das abgetastete Bild lässt sich dann so schreiben:

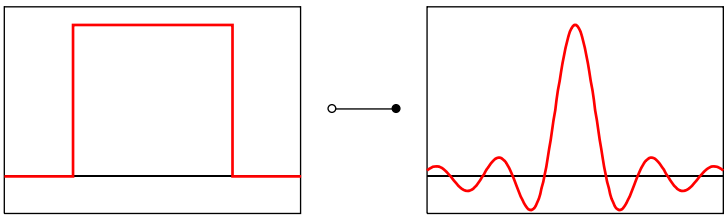
$$g_s(\vec{x}) = g_{\text{CCD}}(\vec{x}) \dot{w}(\vec{x}) \cdot \cdot \sum_{m,n} \delta(\vec{x} - \vec{r}_{m,n})$$

Zusätzlich multipliziert man noch eine **„Fensterfunktion“** $w(\vec{x})$ an das Signal, die nur innerhalb der endlichen Größe des Bildsensors (z.B. $N = L \times L = 1024 \times 1024$ Pixel) 1 ist und sonst 0. Damit hat das Bild eine fest vorgegebene Länge/Breite L und wird mit einer Gitterkonstante Δx gesampelt.

Diese Abtastung lässt sich auch im Fourier-Raum betrachten: Dort kann man nur Frequenzkomponenten zwischen $\lambda_{\text{min}} = \frac{1}{L/2}$ und $\lambda_{\text{max}} = \Delta x$ darstellen (siehe Abtasttheorem). Dabei ist $L = N$ die Länge des Bildausschnittes (z.B. $L = 1024$ Pixel) und Δx das Abtastungs-Intervall (z.B. $\Delta x = 1$ Pixel). Diese minimalen und maximalen Wellenlängen entsprechen maximalen und minimalen Wellenzahlen $k_{\text{max}} = 1/\lambda_{\text{min}}$ und $k_{\text{min}} = 1/\lambda_{\text{max}}$. Im Fourier-Raum, wie im Realraum werden nur genau N Bildpunkte benutzt, sodass man im Fourier-Raum wieder ein Gitter, mit der Schrittweite $\Delta k = \frac{k_{\text{max}} - k_{\text{min}}}{N} = \frac{}{\Delta x}$ erhält. Man kann also ebenso nur eine diskrete, endliche Menge von Frequenzkomponenten darstellen, wie man nur eine diskrete und endliche Menge von Bildpunkten hat.

Die Multiplikation mit einem Gitter im Realraum entspricht nach dem Faltungstheorem einer Faltung mit einem Gitter im Fourier-Raum. Dabei benutzt man, dass die Fourier-Transformierte eines δ-Gitters wieder ein solches Gitter ist. Dies bedeutet aber, dass man die Fourier-Transformierte durch das Sampling, mit einem Delta-Kamm faltet. Dies führt dazu, dass sich das Muster an jedem Punkt dieses Gitters wiederholt. Man erhält also im Fourier-Raum nicht nur eine Fourier-Transformierte des Bildes, sondern unendlich viele, die alle nebeneinander liegen. Dies kann man auch so einsehen: Die Fourier-Reihe war nur für periodische Strukturen auf einem endlichen Intervall definiert, während die Fourier-Transformation auf beliebige Signale auf einem unendlich großen Intervall anwendbar ist. Durch die Überführung der kontinuierlichen Fourier-Trafo in die diskrete DFT, bringt man wieder die Begrenzung des Intervalles mit ein, was dazu führt, dass die beschriebenen Strukturen periodisch sein müssen, wie bei der Fourier-Reihe.

Die Multiplikation mit der Fenster-Funktion entspricht im Fourier-Raum einer Faltung mit deren Fourier-Transformierter $\hat{w}(\vec{k})$. Dies bewirkt die oben erklärte Bandbegrenzung des Signals. Die folgende Abbildung zeigt eine Rechteckfenster und seine Fourier-Transformierte $\mathrm{sinc}(x) = \sin(x)/x$:



Insgesamt bedeutet also die Abtastung mit einem Gitter eine obere und die Fensterfunktion eine untere Frequenzschranke für das Signal.

Standardabtastung: Die oben beschriebene Abtastung, wie in einer CCD-Kamera (jedem Bildpunkt wird der Mittelwert des zugehörigen Pixels zugewiesen) wird als Standardabtastung bezeichnet.

Überabtastung: Um Abtastfehlern zu entgehen wählt man oft (bei bekannter Periodizität der erwarteten Strukturen) ein Abtastgitter, das kleiner ist als die kleinste Struktur. Ist das Gitter zu fein, so steigt der Speicherverbrauch sehr stark, ist es zu grob kommt es zu obigen Fehlern. Üblicherweise wählt man Δx so, dass man etwa 3-6 Abtastpunkte pro Schwingung erhält.

Rekonstruktion des Bildes: Es fehlt noch eine Möglichkeit, um aus dem diskretisierten Bild das kontinuierliche Bild zu rekonstruieren. Dazu muss man die Werte an Punkten zwischen zwei Gitterpunkten aus den umliegenden Grauwerten interpolieren. Üblicherweise hängt dann der Grauwert mit abstandsabhängig von den umliegenden Grauwerten ab.

4.2 Quantisierung

Der letzte Schritt bedeutet die Überführung des kontinuierlichen Signals in ein diskretes Signal, wie es etwa mit den Datentypen `int` oder `char` dargestellt werden kann. Dies gescheit in sog. Analog-/Digitalwandlern, elektronischen Bauelementen, die ein Eingangssignal mit verschiedenen vorgegebenen Signalstufen vergleichen und einen entsprechenden Digitalwert ausgeben, z.B.:

$$
\begin{aligned}
I = 0 \quad &\rightarrow 0 \\
&\ldots \\
I = 0.5 \quad &\rightarrow 127 \\
&\ldots \\
I = 1 \quad &\rightarrow 255
\end{aligned}
$$

Die folgende Abb. 4.4 zeigt ein und dasselbe Bild, aber mit verschiedenen Anzahlen an Grau- und Farbstufen quantisiert.

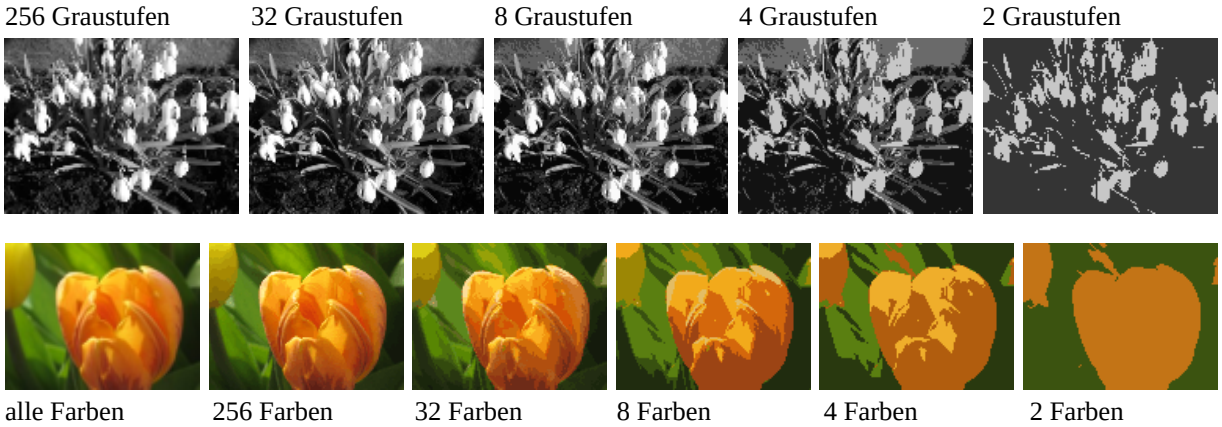


Abb. 4.4: Quantisierung eines Bildes mit unterschiedlich vielen Grau- und Farbstufen

Üblicherweise sind die Quantisierungsstufen in der Bildverarbeitung äquidistant. Um andere Stufen zu erreichen kann man am einfachsten das Bild vor der Quantisierung transformieren. Treten z.B. sehr große Intensitäten $I \approx 1$ und gleichzeitig sehr kleine Intensitäten $I \approx 0.005$ auf, wobei in beiden Bereichen Details aufgelöst werden sollen, so kann man vor der Quantisierung den Logarithmus der Intensitäten an den Bildpunkten berechnen. Dies führt dann bei Anwendung einer äquidistanten Quantisierung zu Grauwerten, die nicht-äquidistanten Intensitäten entsprechen.

Macht man ein einziges Bild, so kann man Grauwerte mit der Genauigkeit Δg messen, die dem halben Abstand zweier Grauwertstufen entspricht. Es gilt also $\sigma \propto \Delta g$. Macht man allerdings mehrere Messungen und mittel so über N Messungen an ein und demselben Pixel, so sinkt die Standardabweichung/der Fehler mit N:

$$\sigma \propto \frac{\Delta g}{\sqrt{N}}$$

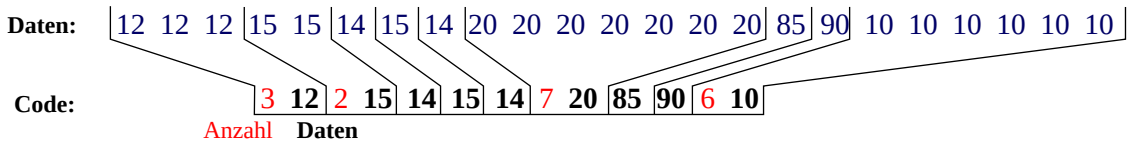
Man kann so durch wiederholte Bildaufnahme die Genauigkeit steigern. Es gibt aber meist eine untere Grenze für die Standardabweichung, die durch andere (systematische) Effekte im Bildaufnahmesystem bedingt sind.

5 Bildrepräsentation

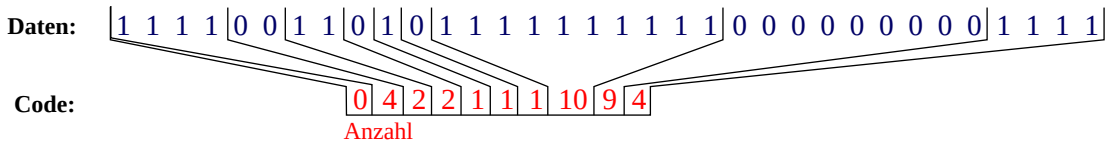
Üblicherweise werden Bilder als Feld im Speicher repräsentiert. Man kann dann mit 1,2,3,... Indizes auf die einzelnen Pixel zugreifen. Manchmal sind aber auch andere Repräsentationen von nutzen:

5.1 Lauflängenkodierung

Grauwertbilder lassen sich vorteilhaft mit der Lauflängenkodierung (*run length encoding*, *RLE*) komprimieren. Dazu speichert man nicht die Abfolge der Pixel (in jeder Zeile) ab, sondern gibt an, wie oft Pixel hintereinander vorkommt. Kommt in einer Bildzeile also der Wert w 25-mail hintereinander, so speichert man nicht 25 mal den Wert w, sondern die Anzahl 25 und einmal den Wert w. Diese Kompressionsart wird in einigen Grafikformaten (z.B. TIFF, TGA, BMP) verwendet und eignet sich natürlich vor Allem für Bilder, in denen auf großen Bereichen der gleiche Grauwert steht. Man muss allerdings beachten, dass bei Grauwertbildern noch ein irgendwie gearteter Marker eingeführt werden muss, um zwischen Anzahl und Wert zu unterscheiden, da sonst alle einzelnen Pixel auf die doppelte Größe (Anzahl 1 + Pixelwert) aufgeblasen werden. Hier ein Beispiel:



Diese Kodierung ist bei Schwarz-Weiß-Bildern besonders effektiv, weil hier nicht der Wert mit abgespeichert werden muss. Denn auf ein schwarzes Gebiet folgt immer ein weißes und umgekehrt. Man nimmt an, dass der erste Pixel 0 ist. Ist er dies nicht, so speichert man einfach 0 schwarze Pixel am Anfang ab. Ein Beispiel:



5.2 Quadtree/Octree

Bei der Repräsentation als Quadtree (Octree für 3D) werden Schwarz-Weiß-Bilder immer wieder in vier Quadranten unterteilt. Ist ein Quadrant homogen gefärbt, so wird seine Farbe gespeichert. Andernfalls wird er weiter unterteilt. In Abb. 5.1 ist ein Beispiel gezeigt.

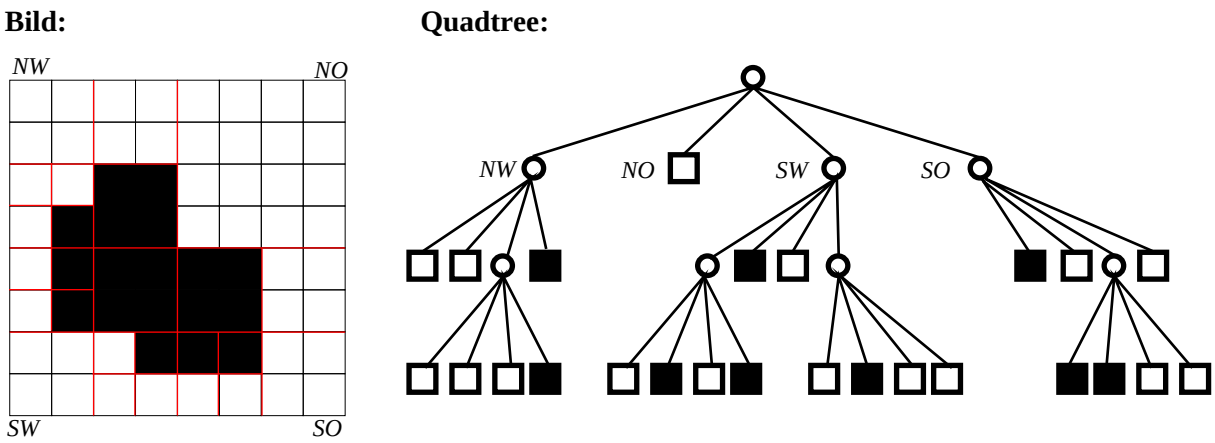


Abb. 5.1: Quadtree eines Bildes

Man kann diesen Baum nach Depth-First-Order durchlaufen und jeden Knoten kodiert abspeichert. **b** steht dabei für schwarz, **w** für weiß und **g** für einen inneren Knoten. Der so entstandene Code ist eindeutig. Die folgenden Zeilen zeigen den Code zum Baum aus Abb. 5.1. In der zweiten Zeile sind zur besseren Lesbarkeit Klammern eingefügt worden:

```
g g w w g w w w b b w g g w b w b b w g w b w w g b w g b b w w w
g( g( w w g( w w w b) b)) w g( g( w b w b) b w g( w b w w)) g( b w g( b b w w) w)
```

Ein solcher Baum ist eine effiziente (platzsparende) Darstellung, wenn viele Blattknoten in den oberen Baumebenen liegen, da diese für große Bildbereiche stehen. Im Gegensatz zur Lauflängekodierung werden hier 2D-Daten kodiert und nicht nur einzelne Zeilen.

5.3 Weitere Kodierungsmethoden

Neben den obigen beiden Varianten wird in Dateiformaten für Grafiken oft der LZW-Algorithmus (Lempel-Ziv-Welch-Algorithmus) eingesetzt. Er ermöglicht es Bilder mit diskreten Grauwerten dann gut zu codieren, wenn sie große gleichfarbige Stellen aufweisen.

Teil II

Grundlegende Operationen auf Bilddaten

6 Punktoperationen und geometrische Operationen

6.1 Punktoperationen

Definition 6.1 (Punktoperation) *Eine Punktoperation $P_{m,n}$ ändert den Grauwert eines Pixels, in Abhängigkeit vom alten Grauwert selbst und evtl. der Position des Bildpunktes. Man kann sie so notieren:*

$$g'_{m,n} = P_{m,n}(g_{m,n})$$

Ist die Punktoperation von der Position des Pixels unabhängig, so nennt man sie homogen. Bei solchen lässt man die Indizes weg:

$$g'_{m,n} = P(g_{m,n})$$

*Hängt eine Punktoperation von zwei Bildern ab, so nennt man sie **dyadische Punktoperation**. Man kann sie so schreiben:*

$$g'_{m,n} = P_{m,n}(g_{m,n}, h_{m,n})$$

Schnelle Berechnung, Lookup-Tables: Vor Allem bei homogenen Punktoperationen treten oft Rechnungen mit den selben Parametern auf. So kann es z.B. nur 256 verschiedene Grauwerte, also auch nur 256 verschiedene Ergebnisse einer homogenen Punktoperation geben. Darum kann es sinnvoll sein alle möglichen vor der Anwendung der Operation Ergebnisse zu berechnen und bei der Anwendung diese nur noch in einer Tabelle nachzushhlagen. Eine solche Tabelle wird auch **Lookup-Table (LUT)** genannt. Liegen etwa die Grauwerte zwischen 0 und 255, so kann sie als einfaches Array implementiert werden, wobei der zu verarbeitende Grauwert als Index dient. Vor Allem bei Punktoperationen mit komplexen arithmetischen Funktionen (wie etwa dem Logarithmus) ergibt dies eine wesentliche Beschleunigung. Es ändert sich dabei zwar nicht die Komplexität der Operation, wohl aber die Vorfaktoren. Eine LUT-Operation ist umso effizienter, je weniger Graustufen ein Bild enthält. In einigen Frame-Grabbern sind LUTs schon in Hardware implementiert und ermöglichen so eine schnelle Vorverarbeitung der eingelesenen Bilder.

Auch dyadische Punktoperationen können mit LUTs implementiert werden. Die LUT ist dann ein 2D-Array.

6.2 Beispiele für Punktoperationen

lineare Spreizung des Grauwertbereiches: Die folgende Abb. 6.1 zeigt was mit einem kontrastarmen Bild passiert, wenn man den Grauwertbereich linear spreizt.

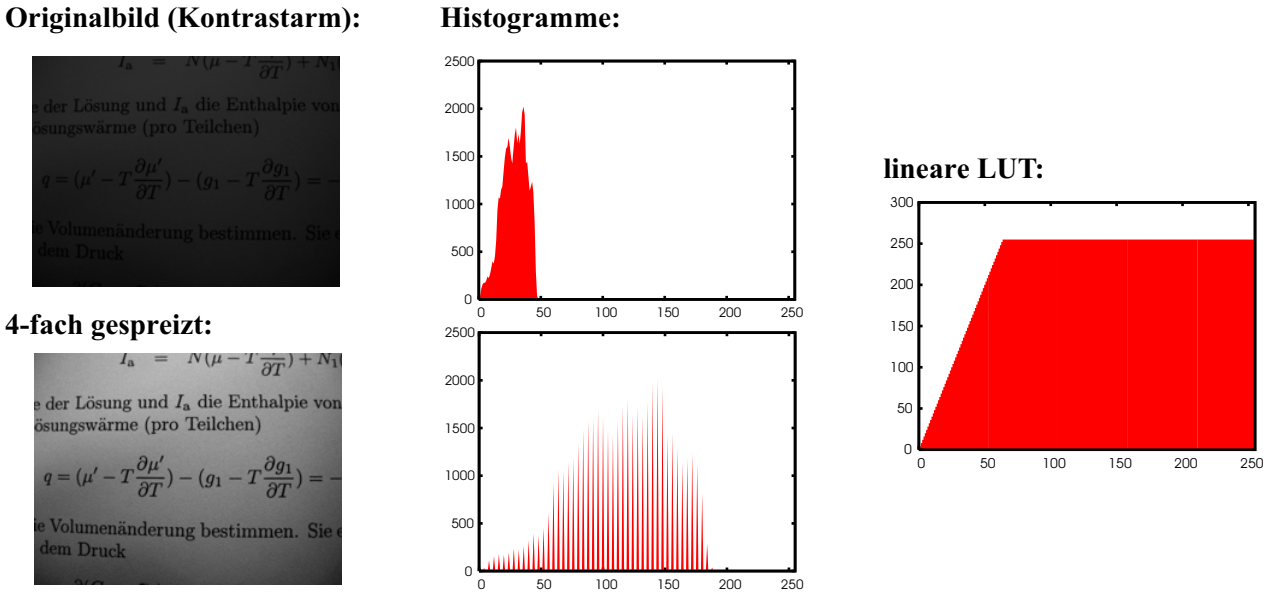


Abb. 6.1: Spreizung des Grauwertbereiches eines Bildes

Jeder Grauwert wird dafür mit einem Faktor α multipliziert. Der Grauwertbereich bleibt aber auf $[0..255]$ beschränkt, sodass im oberen Bereich ein Abschneiden erfolgt. Evtl. kann man auch noch eine Verschiebung g_0 einbauen. Man kann diese homogene Punktoperation dann so notieren:

$$S(g) = \alpha \cdot (g - g_0)$$

Das Beschneiden auf den Bereich $[0..255]$ ist hier nicht dargestellt. In C könnte man eine solche LUT so erzeugen:

Listing 6.1: Spreizung des Grauwertbereiches mit Hilfe einer LUT

```
1  unsigned char luts[256], pic[256][256];
2
3  /* Berechnung der LUT */
4  for (int i=0; i<256; i++) {
5    luts[i]=max(0,min(i*4,255));
6  }
7
8  /* Anwendung auf ein 2D-Graustufen-Bild pic */
9  for (int i=0; i<256; i++) {
10   for (int i=0; i<256; i++) {
11     pic[i][j]=luts[pic[i][j]];
12   }
13 }
```

nicht-lineare Spreizung des Grauwertbereiches: Die folgende Abb. 6.2 zeigt was mit einem kontrastarmen Bild passiert, wenn man den Grauwertbereich mit einer γ-Transformation spreizt.

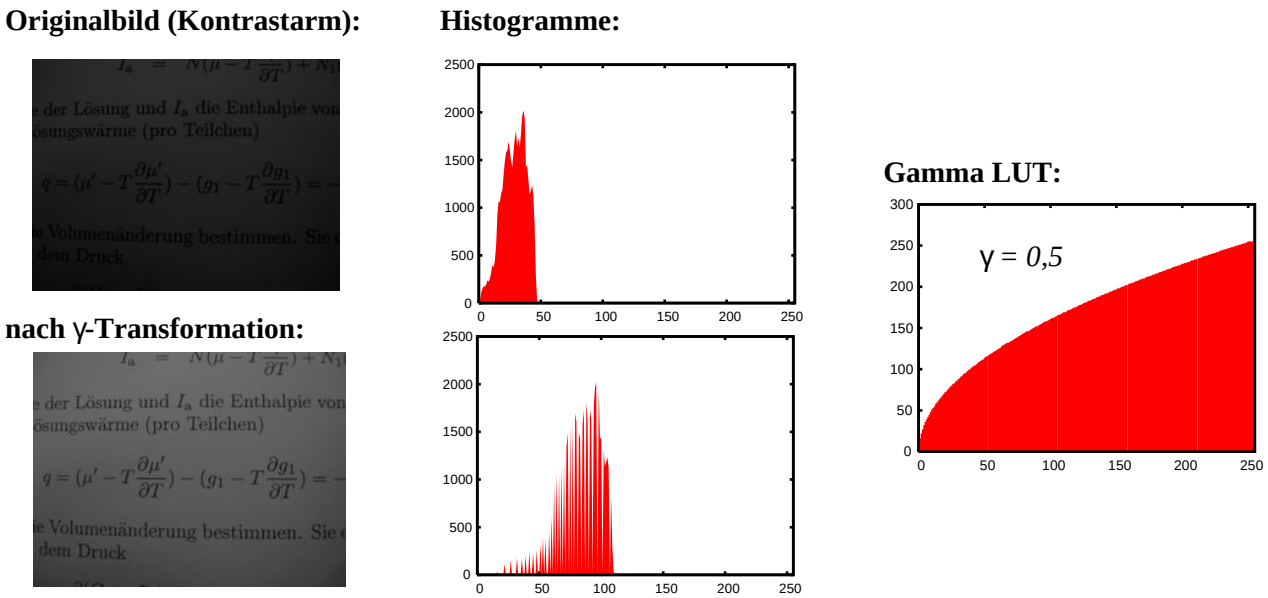


Abb. 6.2: Spreizung des Grauwertbereiches mit einer γ-Transformation

Jeder Grauwert wird dafür folgendermaßen mit dem Parameter γ transformiert:

$$\Gamma(g) = 255 \cdot \left(\frac{g}{255}\right)^{\gamma}$$

Es handelt sich wieder um eine homogene Punktoperation. Die Implementation kann genauso wie oben erfolgen. Eine Überschreitung des Bereiches $[0..255]$ ist nicht möglich. Diese Operation ermöglich eine Vergrößerung des dynamischen Bereiches bei dunklen Grautönen auf Kosten der hellen Grautöne (entspricht dem Verhalten des menschlichen Auges). Siehe dazu auch Abschnitt 2.3.

Bildmittelung: Oft hat man es mit Bilder zu tun, bei denen das Signal ähnlich groß ist wie das Hintergrundrauschen. Dies ist z.B. bei astronomischen Aufnahmen sehr dunkler Sterne der Fall. In solchen Fällen hilft es mehrere Aufnahmen des selben Motives zu machen und diese dann Punktweise zu mitteln. Angenommen es liegen K Aufnahmen $g_{nm}^{(k)}$ vor, dann gilt:

$$g'_{mn} = M_{mn}(g_{nm}^{(1)}, ..., g_{nm}^{(K)}) = \frac{1}{K} \cdot \sum_{l=1}^{K} g_{nm}^{(k)}$$

Nach den Gesetzen der Statistik reduziert sich dabei der Rauschpegel um den Faktor $\frac{1}{\sqrt{K}}$. Die folgende Abb. 6.3 zeigt den Effekt der Mittelung.

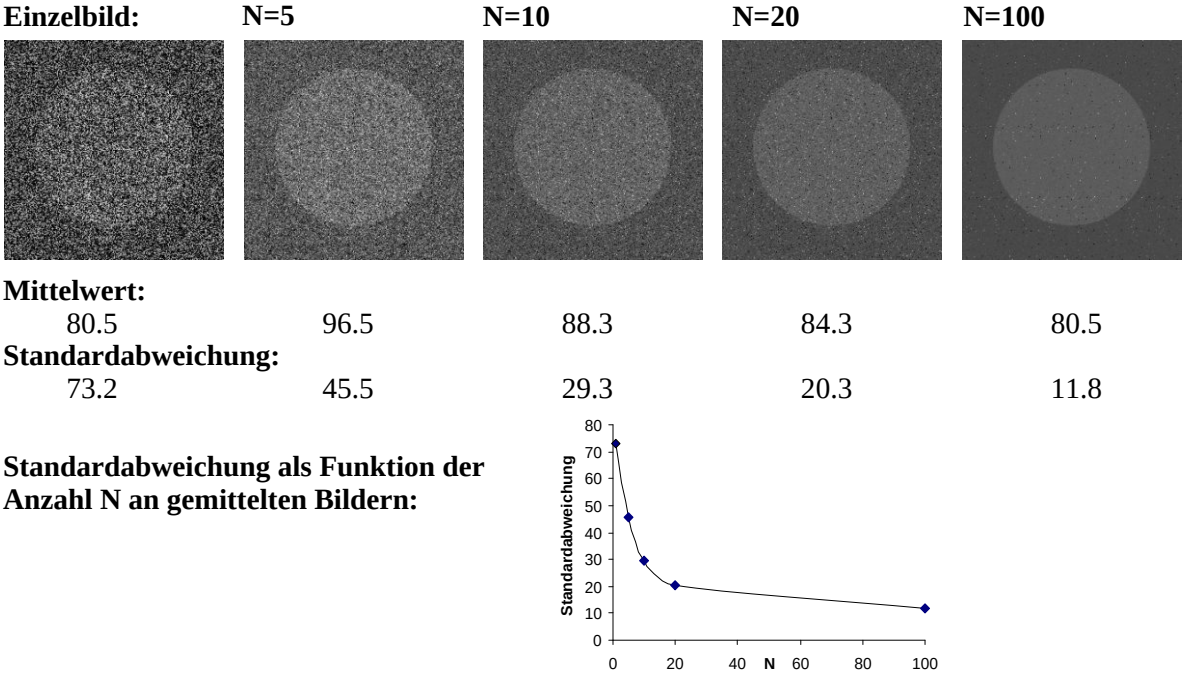


Abb. 6.3: Effekt der Mittelung von N verrauschten Bildern

radiometrische Zweipunkt-Korrektur: Oft macht man Aufnahmen mit ungleichmäßiger Ausleuchtung. Außerdem können die Pixel der Kamera unterschiedlich auf die selbe Intensität reagieren. Dies wird z.B. durch Produktionsfehler und durch Staub oder Wassertröpchen auf dem (gekühlten) CCD-Chip hervorgerufen. Man kann solche Fehler aber korrigieren. Dazu macht man zwei Referenzaufnahmen. Ein sog. **Dunkelbild („darkfield“)** d_{mn} und ein Referenzbild r_{mn} ohne Objekt. Die obigen Effekte kann man mit folgender inhomogener Punktoperation korrigieren:

$$g'_{mn} = R_{mn}(g_{nm}, r_{mn}, d_{mn}) = \text{const} \cdot \frac{g_{nm} - d_{mn}}{r_{nm} - d_{mn}}$$

In der folgenden Abbildung Abb. 6.4 sieht man ein Beispiel für die Anwendung der Kalibrierung, allerdings ohne ein Dunkelbild, da hier Störungen durch das Abbildungssystem ausgeglichen werden sollen, die Belichtung aber weit über dem Rauschlevel lag.

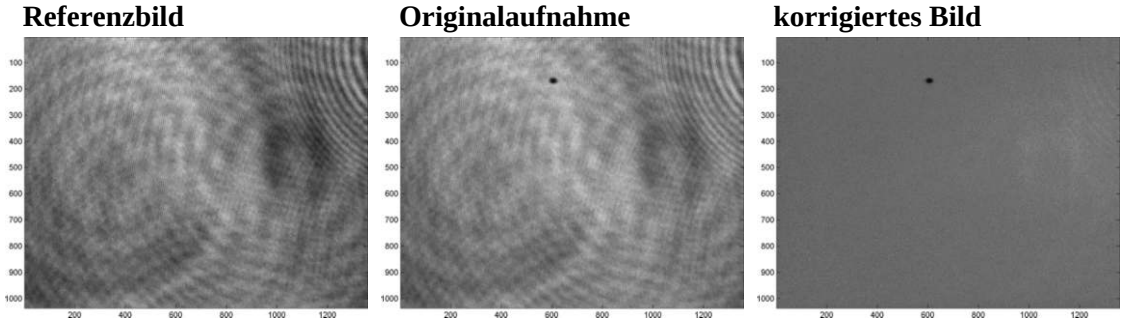


Abb. 6.4: Kalibrierung des Absorbtionsbildes eines Bose-Einstein-Kondensates (dunkler Fleck) mit einem Referenzbild, aber ohne Dunkelbild ($d_{mn} = 0$). Die Ringe im Referenz- und Originalbild sind die die Störung durch das Abbildungssystem, die herausgefiltert werden sollen.

Die Zweipunkt-Korrektur ist nicht immer passend. Manchmal ist es nötig nicht-lineare Störungen zu kalibrieren. Man nimmt dann z.B. drei Bilder auf und korrigiert mit einer parabolischen Kurve, oder Kurven noch höherer Ordnung (mit entsprechend mehr Referenzbildern).

Fensterfunktionen: Bevor man die Fourier-Transformierte eines Bildes berechnet, muss man das Bild mit einer Fensterfunktion multiplizieren, die dafür sorgt, dass die Grauwerte zum Rand hin auf 0 abfallen. Ist dies nicht gegeben, so wirkt die wirkt die Begrenzung des Bildes wie die Faltung mit einer Rechteck-Funktion und das Spektrum des Bildes wird durch die zugehörige sinc-Funktion gestört. Oft verwendet man ein Sinus-Fenster, da es nur eine einzige Komponente im Fourier-Raum hat.

$$g_{mn} = W_{mn}(g_{mn}) = g_{mn} \cdot \sin\left(\frac{\pi m}{M}\right) \cdot \sin\left(\frac{\pi n}{N}\right)$$

In der folgenden Abb. 6.5 wird der Effekt illustriert.

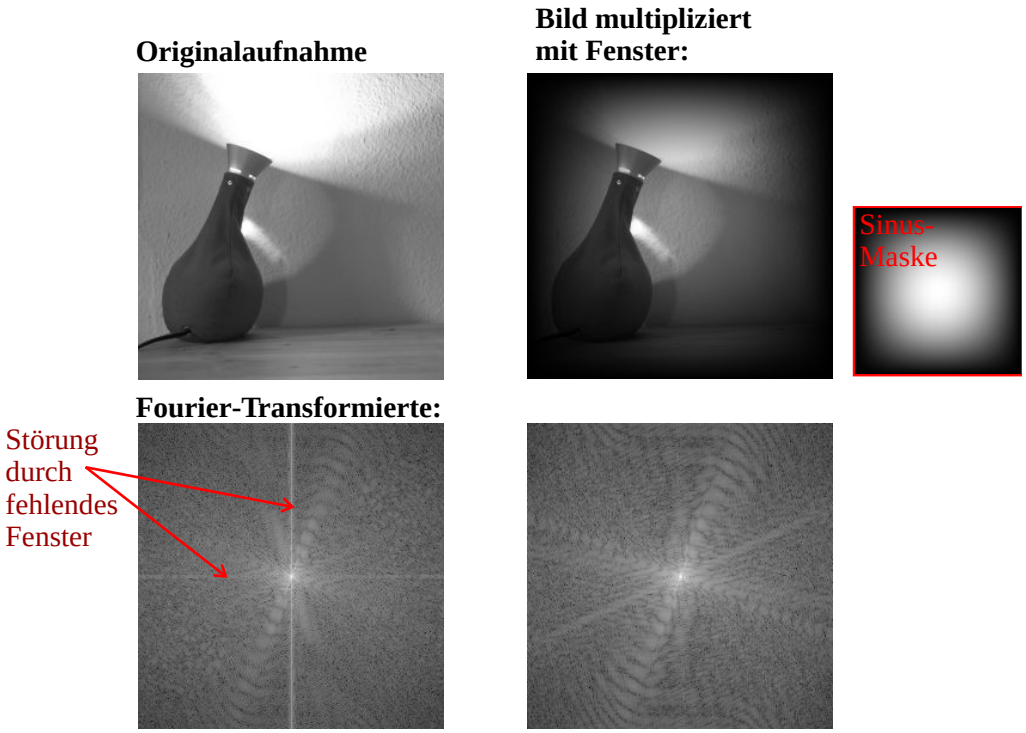


Abb. 6.5: Effekt eines Sinus-Fensters auf die FFT

6.3 Geometrische Transformationen

Es gibt eine weitere Klasse von Punktoperationen, sog. geometrische Transformationen. Sie ändern in erster Linie nicht den Grauwert eines Pixels, sondern seine Position:

$$\vec{x}' = M(\vec{x})$$

Damit die Pixel den Bildes g_{mn} nach der Transformation wieder auf das Gitter eines Pixelbildes passen ist es evtl. nötig an Zwischenstellen zu interpolieren. In Abb. 6.6 sind einige mögliche Transformationen gezeigt. Viele dieser Operationen können als einfache lineare Abbildungen dargestellt werden, wenn man sog. homogene Koordinaten verwendet.

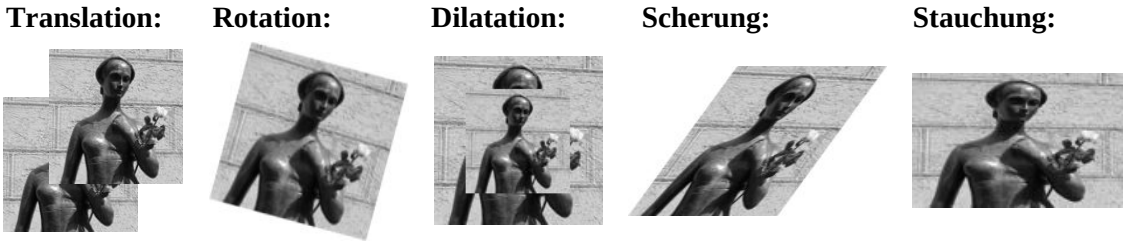


Abb. 6.6: verschiedene geometrische Transformationen

7 Nachbarschaftsoperationen

Definition 7.1 (Nachbarschaftsoperation) *Eine Nachbarschaftsoperation ändert den Grauwert g_{mn} eines Pixels in Abhängigkeit von ihm selbst und in Abhängigkeit der Grauwerte in einem kompakten Gebiet rund um g_{mn}. Man notiert solche Operationen mit Operatoren $\mathcal{A}, \mathcal{B}, \ldots$:*

$$\vec{g}' = \mathcal{A}\,\vec{g}$$

Man kann verschiedene Nachbarschaftsoperationen unterscheiden, die in den folgenden Abschnitten beschrieben werden:

- lineare verschiebungsinvariante Filter (LSI), siehe 7.1
- Rangordnungsfilter, siehe 7.2
- rekursive Filter, siehe 7.3

7.1 lineare verschiebungsinvariante Filter

7.1.1 Faltungen

LSI-Filter sind Filter, die den Grauwert basierend auf einer kleinen Umgebung verändert. Dazu werden die Grauwerte in der Umgebung gewichtet und aufsummiert. Diese mathematische Operation wird als Faltung bezeichnet:

Definition 7.2 (Faltung) *Man betrachte ein diskretes 1D- bzw. 2D-Bild $g_{m(n)}$ und eine Faltungsmaske h_{mn} mit $m, n \in [-r, r]$. Es gilt:*

$$\textbf{\textit{1D:}} \quad g'_n = \sum_{k=-r}^{r} h_k \cdot g_{n-k} \qquad\qquad \textbf{\textit{2D:}} \quad g'_{mn} = \sum_{k=-r}^{r} \sum_{l=-r}^{r} h_{kl} \cdot g_{m-k,n-l}$$

Diese Operation lässt sich auch für kontinuierliche Signale formulieren:

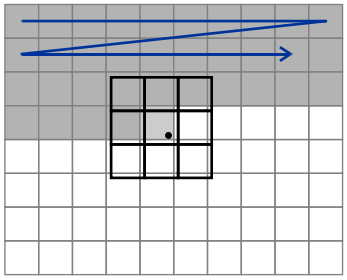
$$\textbf{\textit{1D:}} \quad g(x) = \int h(x') \cdot g(x - x')\,\mathrm{d}x' \qquad\qquad \textbf{\textit{2D:}} \quad g(\vec{x}) = \iint h(\vec{x}') \cdot g(\vec{x} - \vec{x}')\,\mathrm{d}^W x'$$

Die folgende Abb. 7.1 illustriert den praktischen Ablauf einer Faltung.

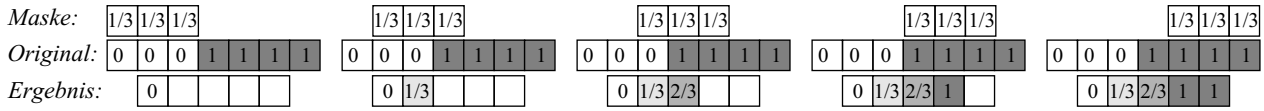
Beispiel für eine Faltungsmaske:

0	1	0
1	-4	1
0	1	0

bildliche Darstellung der Faltung:



Beispiel für eine 1D-Faltung:



Beispiel für eine 1D-Faltung auf Zwischengitter:

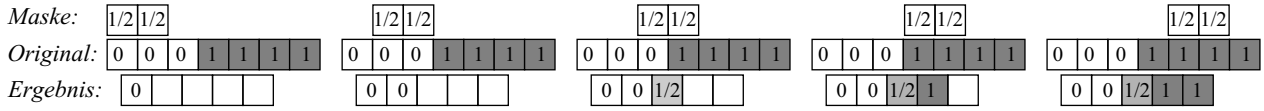


Abb. 7.1: Beispiel einer Faltungsmaske (Laplace-Filter) und anschauliche Darstellung der Faltungsoperation

Die Koeffizienten einer Faltungsmaske werden üblicherweise in der obigen Schreibweise dargestellt. Dabei ist zu beachten, dass die Summe über die Koeffizienten 1 ergeben muss. Die Koeffizienten werden aber üblicherweise als ganze Zahlen dargestellt, sodass man evtl. noch einen Normierungsfaktor berücksichtigen muss. Wird dieser Faktor fortgelassen, so wird der gegebene Grauwertbereich evtl. überschritten.

Die Maske wird über das Bild geschoben. An der aktuellen Position werden die Bildpunkte unter der Maske mit dem entsprechenden Faktoren multipliziert. Danach werden alle diese Werte aufaddiert. Deswegen bezeichnet man das Verfahren als **Wichten und Addieren**.

Die bisher aufgeführten Faltungsmasken haben ungeradzahlige Breiten $2r + 1$. Ist die Breite einer Faltunsgmaske geradzahlig, so liegt das Ergebnisbild auf einem Zwischengitter (siehe Abb. 7.1).

Symmetrien: Es gibt zwei mögliche Symmetrien bei diskreten LSI-Filtern:

- **gerade Symmetrie:** $h_{-m,n} = h_{m,n}$ und $h_{m,-n} = h_{m,n}$
- **ungerade Symmetrie:** $h_{-m,n} = -h_{m,n}$ und $h_{m,-n} = -h_{m,n}$

Nutzt man diese Symmetrien aus, so kann man die Berechnung der Filter vereinfachen, da jetzt die Summe nur noch über die halbe Maske laufen muss. Außerdem spart man dabei Multiplikationen. Eine tatsächliche Verbesserung der Rechenzeit bringt dies nur dann, wenn Multiplikationen teurer sind, als Additionen.

Randeffekte: Am Rand eines Bildes gibt es bei der Faltung Probleme: Wie soll man die fehlenden Pixel behandeln? Mögliche Lösungen sind:

1. Für die Randpixel werden keine neuen Bildpunkte berechnet. Der Nachteil dabei ist, dass das Bild kleiner wird. Bei der Anwendung mehrerer Filter hintereinander wird das Bild somit kleiner und kleiner.
2. **alles Null:** Hierbei werden alle Randpixel auf 0 gesetzt. Dies führt allerdings evtl. zu großen Fehlern, wenn sich die Randpixel des Bildes von 0 unterscheiden. Dafür ist diese Methode am einfachsten zu implementieren.

3. **zyklische Fortsetzung:** Das Bild wird am linken Rand mit dem rechten Bildrand und oben mit dem unteren Bildrand fortgesetzt (Aufrollen des 2D-Bildes zu einem Torus im 3D-Raum). Dabei ergibt sich eine sog. zyklische Faltung. Wird die Faltung durch eine Multiplikation im Fourier-Raum ausgeführt, so entspricht sie dieser zyklischen Faltung. Durch die periodische Fortsetzung kann es am Bildrand zu Grauwertsprüngen kommen, die nach Anwendung der Filtermaske zu ungewollten Artefakten führen. Darum wird diese Lösung selten verwendet. Sie ist unproblematisch, wenn das Bild zum Rand hin überall auf den gleichen Grauwert abfällt.
4. **gespiegelter Rand:** Das Bild wird gespiegelt fortgesetzt. Diese Randbedingung führt schon zu recht wenigen Fehlern.

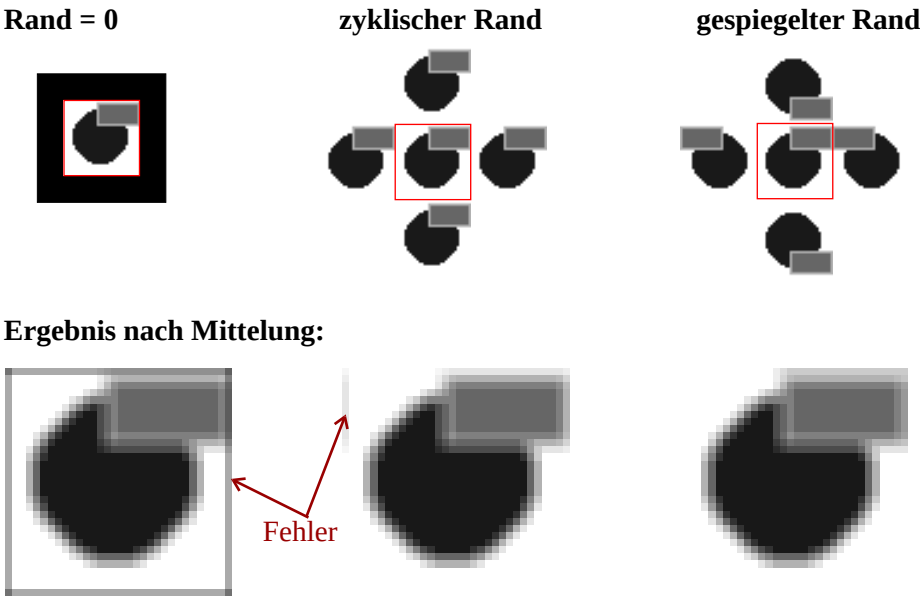


Abb. 7.2: Prinzip verschiedener Randbedingungen und die Fehler, die dabei auftreten können. Es wurde ein 3×3 Mittelwert-Filter eingesetzt

Keine der angegebenen Methoden kann eine fehlerfreie Faltung garantieren. Darum sollte man darauf achten, dass die interessanten Objekte im Bild mindestens eine halbe Maskenbreite vom Rand entfernt liegen.

Probleme bei der Implementierung von Faltungsalgorithmen: Neben den Randbedingungen treten noch weitere Probleme auf:

- Mit steigender Dimension nimmt die Komplexität des Faltungsalgorithmus und die Größe der Nachbarschaft schnell zu.
- Bei berechnen der Faltung müssen in der einfachsten Implementierung die Ergebnisse in ein neues Bild geschrieben werden, da überschriebene Werte evtl. noch gebraucht werden (siehe Abb. 7.1, rechts oben).

7.1.2 Eigenschaften von LSI-Filtern

Satz 7.1 (Linearität) *Jeder LSI-Operator (Filter) $\mathcal{H}$ ist linear. D.h. zusammen mit zwei Bildern $\vec{f}, \vec{g} \in \mathbb{C}^W$ und zwei Skalaren $a, b \in \mathbb{C}$ gilt:*

$$\mathcal{H}\big(a\,\vec{g} + b\,\vec{f}\big) = a \cdot (\mathcal{H}\vec{g}) + b \cdot (\mathcal{H}\vec{f})$$

Satz 7.2 (Verschiebungsinvarianz) *Jeder LSI-Operator (Filter) $\mathcal{H}$ ist verschiebungsinvariant. Dies bedeutet, dass der Operator nicht von der Position im Bild abhängt, sich also während der Faltung nicht verändert. Mit dem* ***Verschiebungsoperator*** *$\mathcal{S}_{mn}$, der ein Bild um $[m,n]^t$ verschiebt, kann man die Verschiebungsinvarianz so formulieren: Jeder Operator, $\mathcal{H}$, der mit dem Verschiebungsoperator $\mathcal{S}_{mn}$ vertauscht, also $\mathcal{H}\mathcal{S}_{mn} = \mathcal{S}_{mn}\mathcal{H}$ ist verschiebungsinvariant. Es ist also egal, ob man den Operator erst anwendet und dann das Ergebnis beliebig verschiebt, oder umgekehrt.*

Satz 7.3 (Kommutativität) *Zwei beliebige LSI-Operator $\mathcal{A}, \mathcal{B}$ sind kommutativ, d.h.:*

$$\mathcal{A}\mathcal{B} = \mathcal{B}\mathcal{A}$$

Die Reihenfolge, in der LSI-Operatoren auf ein Bild angewendet werden ist also egal.

Der Beweis kann einfach im Fourier-Raum erfolgen, weil dort Faltungen in Multiplikationen übergehen, die bekanntlich kommutativ sind.

Satz 7.4 (Assoziativität) *Zwei beliebige LSI-Operator $\mathcal{A}, \mathcal{B}$ sind assoziativ, d.h.:*

$$\mathcal{A}\mathcal{B} = \mathcal{C}$$

Dabei ist $\mathcal{C}$ ebenfalls ein LSI-Filter. Die Anwendung zweier (oder mehrerer) LSI-Filter kann zu einem neuen LSI-Filter zusammengefasst werden.

Satz 7.5 (Distributivität über die Addition) *Zwei beliebige LSI-Operator $\mathcal{A}, \mathcal{B}$ sind distributiv bzgl. der Addition:*

$$(\mathcal{A} + \mathcal{B})\vec{g} = \mathcal{C}\vec{g}$$

Dabei ist $\mathcal{C}$ ebenfalls ein LSI-Filter und $\vec{g}$ ein Bild.

Punktantwort: Im Abschnitt 3 über die Fourier-Darstellung von Bildern wurde erklärt, dass ein Bild als Linearkombination von Einzelbildern dargestellt, die jeweils nur einen Punkt enthalten, der von 0 verschieden ist. Ein solches Bild entspricht im kontinuierlichen Raum einer δ-Funktion. Ein solches Basisbild sei mit $\vec{p}_{mn}$ bezeichnet. Es hat an der Stelle $(m,n)^t$ den Pixel $\neq 0$. Ein beliebiges Bild $\vec{g}$ lässt sich dann so darstellen:

$$\vec{g} = \sum_m \sum_n g_{mn}\vec{p}_{mn}$$

Da jeder LSI-Operator $\mathcal{H}$ linear ist, gilt weiter:

$$\mathcal{H}\vec{g} = \sum_m \sum_n g_{mn}(\mathcal{H}\vec{p}_{mn})$$

Es ist also interessant, wie die Antwort eines Filters auf einen δ-Puls aussieht. Diese nennt man Impulsantwort.

Definition 7.3 (Impulsantwort) *Die Impulsantwort eines LSI-Filters $\mathcal{H}$ ist das Ergebnis der Anwendung des Filters $\mathcal{H}$ auf ein Impulsbild $\vec{p}_{mn}$, bzw. eine δ-Funktion. Für diskrete Filter ergibt sich als Ergebnis die Filtermaske selbst.*

Da bei diskreten Filtern die Punktantwort endlich ausgedehnt ist (endlich große Filtermaske) spricht man auch von **FIR-Filter** oder **Finite-Impulse-Response-Filtern**.

Transferfunktion: Die Faltung im Realraum geht im Fourierraum in eine punktweise Multiplikation über. Es ist also auch interessant sich die Fourier-Transformierte der Punktantwort $\vec{h}$ eines LSI-Filters $\mathcal{H}$ anzusehen. Obiges lässt sich so zusammenfassen::

$$\mathcal{H}\vec{g} = \vec{h} \circledast \vec{g} = \mathcal{F}^{-1}\left[\hat{\vec{h}} \cdot \hat{\vec{g}}\right]$$

Man definiert so:

Definition 7.4 (Transferfunktion) *Die Impulsantwort eines LSI-Filters $\mathcal{H}$ sei mit $\vec{h}$ bezeichnet. Ihre Fourier-Transformierte $\hat{\vec{h}}$ bezeichnet man als Transferfunktion (TF). Sie gibt an, wie stark welcher Frequenzanteil im endgültigen Bild enthalten ist.*

Im Allgemeinen ist die Transferfunktion eine komplexe Größe, sodass bestimmte Frequenzanteile nicht nur skaliert, sondern auch verschoben werden können (sieh Abschnitt 3.6.1 und dort die Sätze über Verschiebungen). Weist ein LSI-Filter Symmetrien auf, so vereinfacht sich dadurch die Transferfunktion. So hat ein reeller Filter mit gerader Symmetrie eine reelle Transferfunktion. Bei ungerader Symmetrie ergibt sich eine rein imaginäre Transferfunktion. Diese Tatsachen ergeben sich direkt aus den Symmetriebeziehungen für die Fourier-Transformation (siehe 3.6.1). Man kann sich weiter überlegen, dass gerade Filter nur Cosinus-Anteile enthalten können, weil diese selber symmetrisch sind. Ungerade Filter zerfallen dagegen nur in Sinus-Anteile. Dies kann man für einen 1D-Filter einfach berechnen. Man geht von der Symmetriebeziehung $h_{-n} = \pm h_n$ aus und rechnet:

$$\begin{aligned}\hat{h}_\nu &= \sum_{n'=-R}^{R} h_{n'} \exp\left(-\frac{2\pi \mathrm{i} n\nu}{N}\right) = \\ &= h_0 + \sum_{n'=1}^{R} h_{n'} \left[\exp\left(-\frac{2\pi \mathrm{i} n\nu}{N}\right) \pm \exp\left(\frac{2\pi \mathrm{i} n\nu}{N}\right)\right]\end{aligned}$$

Daraus ergibt sich schließlich analog zu obigen Behauptungen unter Verwendung von $\mathrm{e}^{\mathrm{i}\varphi} = \cos\varphi + \mathrm{i}\sin\varphi$:

$$\text{gerade} \qquad \hat{h}_\nu = h_0 + 2 \cdot \sum_{n'=1}^{R} h_{n'} \cos\left(\frac{2\pi \mathrm{i} n\nu}{N}\right) \tag{7.1.1}$$

$$\text{ungerade} \qquad \hat{h}_\nu = -2\mathrm{i} \cdot \sum_{n'=1}^{R} h_{n'} \sin\left(\frac{2\pi \mathrm{i} n\nu}{N}\right) \tag{7.1.2}$$

Diese Transferfunktionen sind abhängig von einer diskreten Variable ν. Man kann durch die Setzung $Nk/2 = \nu$ eine neue, kontinuierliche Variable k einführen, die zur besseren Darstellung der Transferfunktion herangezogen werden kann. Sie liegt im Bereich $[-1, 1[$

Zerlegung eines Operator in einfachere Operatoren: Für viele komplexe Faltungsmasken lässt sich eine einfache Zerlegung angeben, die eine schnellere Implementierung erlaubt. So lässt sich etwa folgender 2D-Operator $\mathcal{B}$ in zwei 1D-Operatoren $\mathcal{B}_x, \mathcal{B}_y$ zerlegen, die über eine Faltung verknüpft sind:

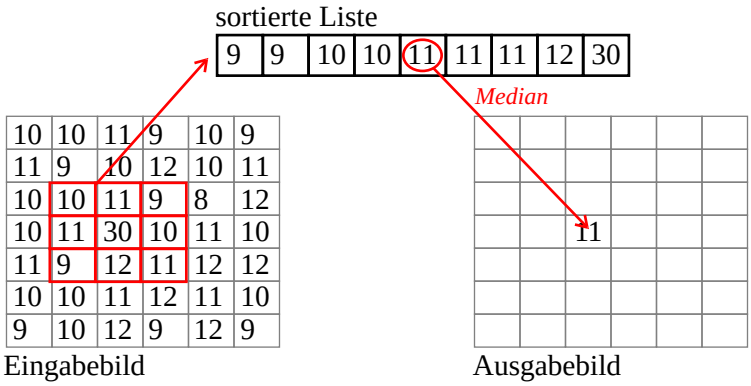
$$\mathcal{B} = \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = (1, 4, 6, 4, 1) \circledast \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} = \mathcal{B}_x \circledast \mathcal{B}_y$$

Man kann also anstatt mit der großen Maske zu falten auch zweimal mit einer kleineren Maske falten. Dies verkleinert die zu betrachtende Nachbarschaft erheblich.

7.2 Rangordnungsfilter

Die LSI-Filter können durch Wichten und Addieren charakterisiert werden. Es gibt aber auch andere Schemata, wie man die Daten einer Umgebung verarbeiten kann. Eine zweite Klasse von (nicht-linearen) Verfahren sind die sog. Rangordnungsfilter. Dabei wird aus den Grauwerte in der Nachbarschaft eine sortierte Liste erstellt. Danach wählt man den neuen Wert des Pixels aus dieser Liste aus. Man nimmt z.B. den Median, das Maximum oder das Minimum. Die folgende Abb. 7.3 veranschaulicht einen Medianfilter.

Funktionsweise eines Medianfilters:



Anwendung eines Medianfilters:

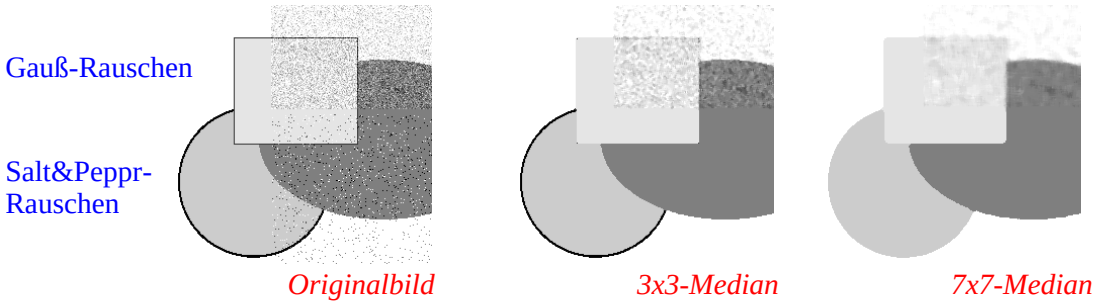
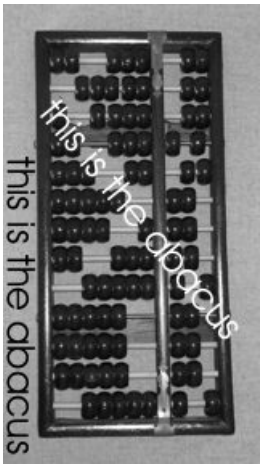


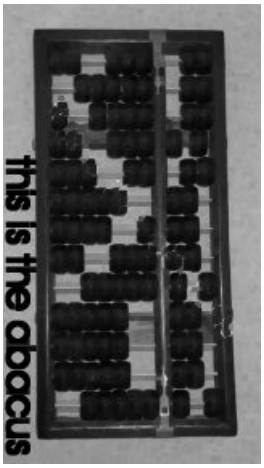
Abb. 7.3: Medianfilter: Prinzip und Anwendung auf ein Testbild

In Abb. 7.3 sieht man unten ein Testbild mit scharfen Kanten verschiedener Dicke. Es enthält zusätzlich rechts oben einen Bereich mit Gauß'schem Rauschen und rechts unten einen Bereich mit Salt-and-Pepper-Rauschen. letzteres ist ein Rauschen, bei dem wenige Peaks stark nach oben oder unten abweichen. Es zeigt sich, dass Medianfilter für die Entfernung solchen Rauschens besonders gut geeignet sind. Es zeigt sich aber auch der negative Einfluss auf scharfe Umrandungen im Bild. So verschwindet schon bei einer kleinen Maske die dünne schwarze Umrandung des Rechtecks und bei einer etwas größeren Maske auch die dicke Umrandung des Kreises. Das Gauß-Rauschen wird nicht sehr erfolgreich geglättet.

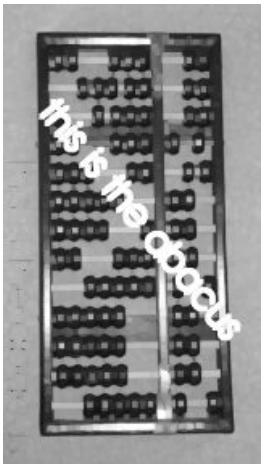
Man kann alternativ auch das Minimum oder Maximum auswählen. In Abb. 7.4 ist die Auswirkung eines solchen Filters gezeigt. Man kann sie benutzen, um z.B. weißen oder schwarzen Text aus einem Bild zu filtern. Ein Minimusfilter vergrößert dunkle Strukturen (kleine Grauwerte) und verkleinert helle Strukturen. So verschwinden in der Abbildung etwa die Glanzlichter und die Perlen den Abakus werden größer. Der Maximumsfilter macht das umgekehrte. Die Glanzlichter werden verstärkt und der weiße Text wird breiter.



Originalbild



Minimum-Filter



Maximum-Filter

Abb. 7.4: Minimums- und Maximums-Filter

7.3 rekursive Filter

Einführung: Es gibt noch eine dritte Nachbarschaftsoperation: Rekursive Filter. Diese greifen explizit auf die Ergebnisse der vorausgegangenen Rechenschritte zurück. Ein einfacher Filter für ein 1D-Signal ist etwa:

$$g'_n = \alpha \cdot g'_{n-1} + (1-\alpha) \cdot g_n$$

Er benutzt explizit den Wert g'_{n-1}, also den Wert, der als letztes „erzeugt“ wurde. Dieser Abschnitt beschränkt sich auf 1D-Filter, da hier alle Prinzipien klar werden, die Notation aber gleichzeitig nicht unnötig kompliziert wird.

Kausalität: Benutzt ein Filter für zeitliche Signale neben dem aktuellen Wert nur „alte“ Werte, so nennt man ihn „kausaler“ Filter. Bei räumlichen Eingabedaten gibt es keine Vorzugsrichtung, wie bei zeitlichen Daten. Dort macht also das Konzept der Kausalität keinen Sinn.

Verallgemeinerung:

Definition 7.5 (kausale rekursive Filter (1D)) *Ein kausales rekursives 1D-Filter bezieht bei der Berechnung des n-ten Pixels/Wertes auch die bereits berechneten Werte mit ein. Außerdem kann es sich auf eine Nachbarschaft beziehen. Dies führt zur folgenden allgemeinen Formulierung, die auf Linearkombinationen der Pixelwerte beschränkt bleibt:*

$$g'_n = \underbrace{-\sum_{k=1}^{S} a_k g'_{n-k}}_{rekursiver\,Anteil} \quad \underbrace{+\sum_{l=-R}^{R} h_l g_{n-l}}_{nichtrekursiver\,Nachbarschaftsanteil} \tag{7.3.1}$$

Punktantwort: Im Gegensatz zu den Filtern aus Abschnitt 7.1 sind die Impulsantworten von rekursiven Filtern meist unendlich ausgedehnt. Es handelt sich also im Allgemeinen nicht um FIR-Filter, sondern um **IIR-Filter (Infinite-Impulse-Response-Filter)**. Dies sieht man leicht an obigem Beispiel. Um die Impulsantwort zu berechnen muss man die Rekursion explizit berechnen, indem man das Signal

$$g_n = \begin{cases} 1 & n = 0 \\ 0 & \text{sonst} \end{cases}$$

einsetzt.

Mit der so berechneten Punktantwort kann man einen nichtrekursiven Filter konstruieren, der zum selben Ergebnis, wie der rekursive Filter führt. Allerdings ist dessen Punktantwort evtl. unendlich ausgedehnt.

Aus obigem Beispiel erhält man:

$$
\begin{aligned}
g'_0 &= \alpha \underbrace{g_{-1}}_{=0} + (1-\alpha) \underbrace{g_0}_{=1} = 1-\alpha \\
g'_1 &= \alpha g'_0 + (1-\alpha) \underbrace{g_1}_{=0} = \alpha \cdot (1-\alpha) \\
g'_2 &= \alpha g'_1 + (1-\alpha) \underbrace{g_2}_{=0} = \alpha^2 \cdot (1-\alpha) \\
&\vdots \\
g'_n &= \alpha g'_{n-1} + (1-\alpha) \underbrace{g_n}_{=0} = \alpha^n \cdot (1-\alpha)
\end{aligned}
$$

Stabilität: Man nennt einen Filter stabil, wenn seine Impulsantwort gegen 0 konvergiert. Um zu ermitteln ob ein rekursives Filter stabil ist muss die Punktantwort rekursiv berechnet werden. Der Filter aus obigem Beispiel ist also stabil für $|\alpha| < 1$.

Transferfunktion: Um die Transferfunktion eines rekursiven Filters zu berechnen geht man von der Gleichung (7.3.1) aus obiger Definition aus. Diese wird etwas umgestellt und dann Fourier-Transformiert. Man erhält dann:

$$
\begin{aligned}
g'_n &= -\sum_{k=1}^{S} a_k g'_{n-k} + \sum_{l=-R}^{R} h_l g_{n-l} \\
\text{mit } a_0 = 1 \sum_{k=0}^{S} a_k g'_{n-k} &= \sum_{l=-R}^{R} h_l g_{n-l} \\
\hat{g}'(k) \cdot \sum_{j=0}^{S} a_j \cdot \exp(-2\pi \mathrm{i} jk) &= \hat{g}(k) \sum_{l=-R}^{R} h_l \exp(-2\pi \mathrm{i} lk)
\end{aligned}
$$

Die Exponentialfunktionen rühren aus den Sätzen über verschobene Signale her, da ein Term der Form $g_{n-n'}$ in der zweiten Zeile nichts anderes als ein um n' verschobenes Signal darstellt. Im Fourier-Raum wird aus der Faltung eine Multiplikation, also gilt $\hat{g}'(k) = \hat{h}(k) \cdot \hat{g}(k)$. Damit ergibt sich zusammen mit obiger Rechnung:

$$
\hat{h}(k) = \frac{\hat{g}'(k)}{\hat{g}(k)} = \frac{\sum_{l=-R}^{R} h_l \exp(-2\pi \mathrm{i} lk)}{\sum_{j=0}^{S} a_j \cdot \exp(-2\pi \mathrm{i} jk)}
$$

Zur weiteren Analyse dieses Ausdruckes verwendet man die sog. z-Transformation.

8 Multiskalenrepräsentation

Oft enthält ein und dasselbe Bild verschiedene Strukturen, die sich auf verschiedenen Längenskalen befinden. Die folgende Abb. 8.1 zeigt Beispiele dafür. Je gröber die Auflösung, desto weniger feine Details sind sichtbar (kleine Punkte in den unteren Bildern verschwinden). Teilweise fließen grobe Strukturen auch ineinander (drei Peaks links in den 1D-Daten werden zu zwei Peaks).

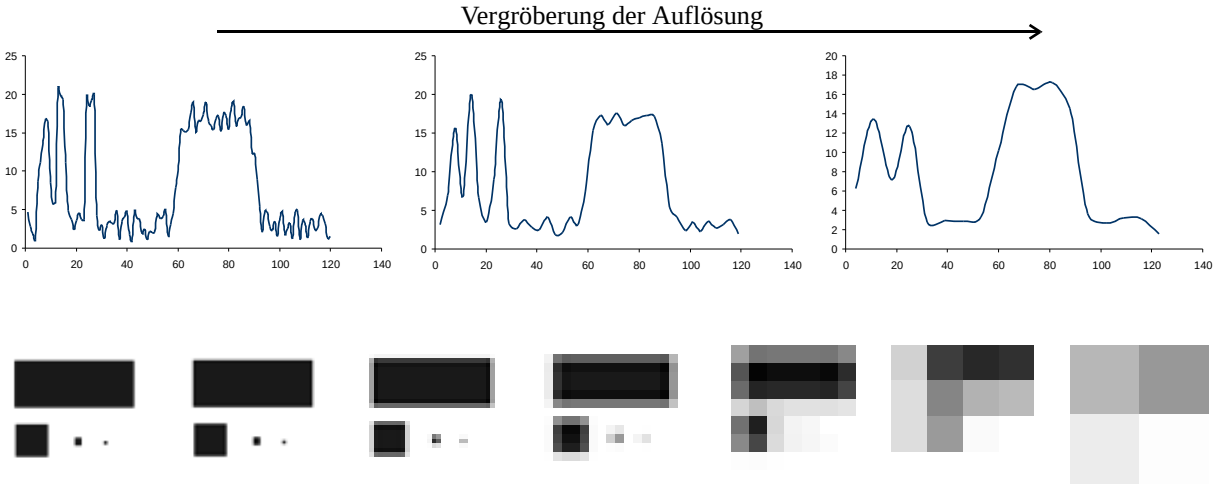


Abb. 8.1: Darstellung von 1D-Daten und Bilddaten in verschiedenen Auflösungen

Man sucht also eine Datenstruktur, die solche multiskalaren Daten repräsentieren kann. Will man z.B. die Grobstruktur eines Bildes bestimmen, so stört evtl. die Feinstruktur nur. Um etwa Peaks zu identifizieren ist wohl der zweite Graph besser geeignet, als der erste. Außerdem reduziert die Reduktion der Auflösung auch die Anzahl der Daten und damit die Größe einer evtl. anzuwendenden Filtermaske. Damit ist die Strukturerkennung bei gröberer Auflösung nicht nur einfacher, sondern evtl auch schneller. Eine Datenstruktur, die dieses leistet ist die Gauß-Pyramide.

Im Ortsraum hat man üblicherweise nur räumliche Informationen. Die Information über enthaltene Wellenzahlen ist über den gesamten Ortsraum verschmiert. Man hätte aber u.U. auch gerne eine Darstellung des Signals, die eine lokale Orts- und Wellenzahlinformation enthält. Man kann sich z.B. eine Serie von Bandpass-gefilterten Bildern mit sinkender Auflösung vorstellen. Dabei enthält jedes Bild nur solche Strukturen, die einem charakteristischen Wellenzahlbereich entsprechen. Eine passende Datenstruktur ist die Laplace-Pyramide.

8.1 Pyramiden-Zerlegungen

8.1.1 Gauß-Pyramide

Die Gauß-Pyramide besteht aus einer Folge von Bildern, deren Länge und Breite sich von Schritt zu Schritt halbiert. Um das Abtast-Theorem nicht zu missachten muss vor dem Halbieren der Auflösung eine Glättung erfolgen. Man bezeichnet den Glättungsoperator mit $\mathcal{B}$. Bei ihm handelt es sich um einen

Gauß-Filter (bei diskreten Bildern ein Binomialfilter). Zusätzlich benötigt man noch einen Operator $\downarrow_2$, der die Auflösung halbiert, also das Down-Sampling übernimmt.

Definition 8.1 (Gauß-Pyramide) *Sei $\mathcal{B}$ ein Glättungsfilter und der Operator $\downarrow_2$ halbiere die Auflösung der Bilddaten. Das Ausgangsbild sei mit $\vec{G}$ bezeichnet. Die Gauß-Pyramide lässt sich dann rekursiv definieren:*

1. *Die unterste Ebene der Pyramide ist das Originalbild:*

$$\vec{G}^{(0)} = \vec{G}$$

2. *Die jeweils nächste Ebene berechnet sich aus der vorherigen durch Glättung und Abtastung:*

$$\vec{G}^{(n+1)} = \downarrow_2 \left(\mathcal{B}\, \vec{G}^{(n)}\right)$$

3. *diese Vorschrift wird ausgeführt, bis das kleinste Bild nur noch aus einem Bildpunkt besteht.*

Der Glättungsfilter als Tiefpassfilter sei so dimensioniert, dass sich die Grenzfrequenz von Ebene zu Ebene halbiert.

Man verliert also von Ebene zu Ebene an Auflösung und es gehen feine Strukturen (hohe Frequenzanteile) verloren. Man erhält eine Serie von Tiefpass-gefilterten Bildern. Der zusätzliche Speicherbedarf der Gaußpyramide im Vergleich zum Originalbild ist relativ klein. Die Gesamtzahl der Bildpunkte in einer Pyramide zu einem W-dimensionalen Bild ist:

$$N := \sharp\vec{G}^{(0)} + \sharp\vec{G}^{(1)} + \sharp\vec{G}^{(2)} + ... = M^W \cdot \left(1 + \frac{1}{2^W} + \frac{1}{2^{2W}} + ...\right) < M^W \cdot \frac{2^W}{2^W - 1}$$

Damit ergibt sich für $W = 2$ und $M = 256$: Das Originalbild enthält $N^{(0)} = M^W = 65536$ Punkte. Die Gesamte Pyramide $N < \frac{4}{3} \cdot N^{(0)}$. Es wird also weniger als ein drittel mehr Speicher benötigt. Die Berechnung der Pyramide ist ebenfalls sehr effektiv, da die Bilder von Stufe zu Stufe kleiner werden und somit die Filterung immer schneller von Statten geht. Die Pyramide ermöglicht eine effiziente Datenverarbeitung auf den höheren Stufen, weil das Bild kleiner wird und die Filtermasken auch kleiner dimensioniert werden können.

8.1.2 Laplace-Pyramide

Die Gauß-Pyramide stellt eine Serie von Tiefpass-gefilterten Bildern dar. Man kann aus ihr die Laplace-Pyramide berechnen. Diese ist eine Serie von Bandpass-gefilterten Bildern. Jede Ebene enthält also nur Wellenzahlen zu einem bestimmten, eng begrenzten Wellenzahlbereich. Zur Berechnung der Gauß-Pyramide werden zwei aufeinanderfolgende Bilder der Gauß-Pyramide subtrahiert. Dazu muss eines vergrößert werden. Dies erledigt der **Expansionsoperator** $\uparrow_2$. Seine Berechnung kann aber recht komplex sein, weil eine Interpolation der fehlenden Daten benötigt wird. Der Index 2 gibt an, dass die Größe des Bildes (Längendimension) verdoppelt wird.

Definition 8.2 (Laplace-Pyramide) *Mit $\uparrow_2$ sei der Expansionsoperator bezeichnet. Man geht von einem originalbild $\vec{G}$ aus, aus dem eine Gaußpyramide mit den Stufen $\vec{G}^{(0)}, ..., \vec{G}^{(M)}$ konstruiert wurde. Daraus kann man die Laplace-Pyramide $\vec{L}^{(m)}$ rekursiv berechnen:*

1. *Die Bilder der obersten Ebene (kleineste Bilder) sind gleich:*

$$\vec{L}^{(M)} = \vec{G}^{(M)}$$

2. *das nächstgrößere Bild ergibt sich durch Vergrößern und Subtrahieren aus der Gauß-Pyramide:*

$$\vec{L}^{(m)} = \vec{G}^{(m)} - \uparrow_2 \vec{G}^{(m+1)}$$

Um das Originalbild wieder herzustellen muss man nur die Bilder der Laplace-Pyramide rekursiv vergrößern und aufaddieren:

$$\vec{G}^{(M)} = \vec{L}^{(M)}, \qquad \vec{G}^{(m-1)} = \vec{L}^{(m-1)} + \uparrow_2 \vec{G}^{(m)} = \vec{L}^{(m-1)} + \uparrow_2 \left(\vec{L}^{(m)} + \uparrow_2 \vec{G}^{(m+1)}\right) = ...$$

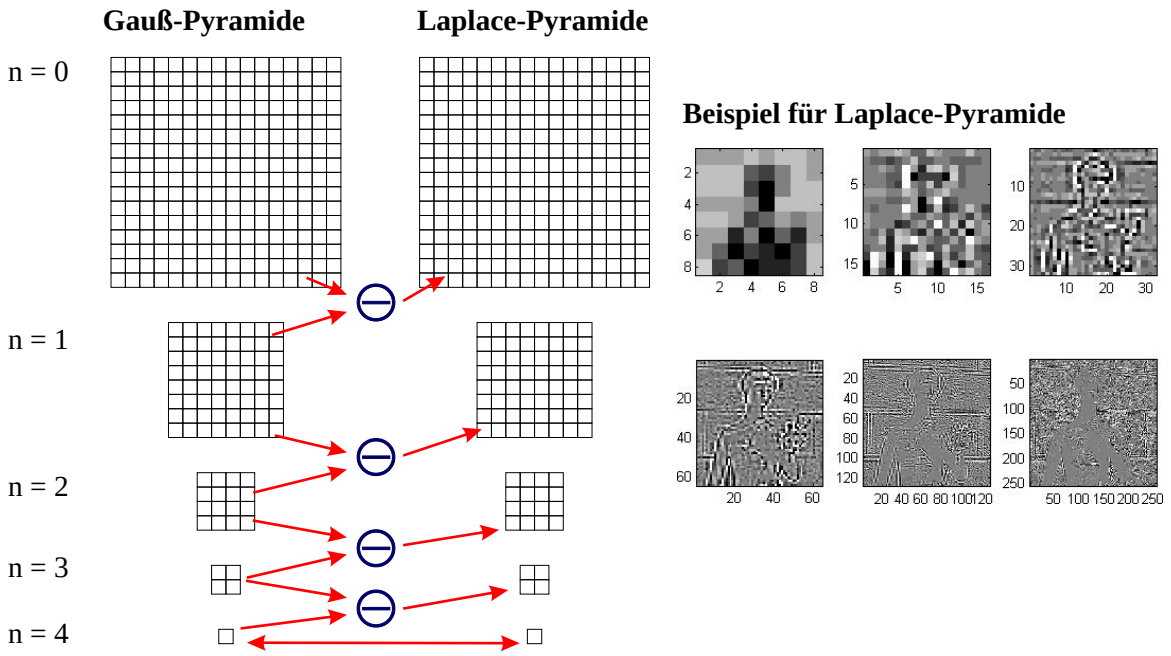


Abb. 8.2: Veranschaulichung der Berechnung der Laplace-Pyramide

8.1.3 Richtungszerlegungen

Man kann die obigen Pyramidenzerlegungen noch um eine Richtungsinformation erweitern. Dabei wird zur Glättung ein separierbarer Filter $\mathcal{B} = \mathcal{B}_x\mathcal{B}_y$ verwendet. Man erhält dann mehrere Bilder pro Stufe, die zusätzlich eine Richtungsinformation enthalten, da die Glättung nur in einer Richtung erfolgte.

8.2 Skalenraumtheorie

Die bisherigen Zerlegungen sind nur Spezialfälle der allgemeinere Skalenraumtheorie. Die Skala eines Bildes charakterisiert die typische Größe von Strukturen im Bild. Bisher wurde die Skala nur in diskreten und recht großen Schritten verändert (immer halbiert). Es wäre aber wünschenswert die Skala kontinuierlich anpassen zu können. Man benötigt dazu einen Prozess, der Bilder kontinuierlich unscharf macht. In

der Physik ist die Diffusion als ein solcher Prozess bekannt. Im folgenden soll der Formalismus der Diffusion auf Bilder angewendet werden, um die Auflösung kleiner Strukturen mit einem Skalenparameter ξ kontinuierlich einstellen zu können.

8.2.1 Diffusionsgleichungen

Zunächst stellt man sich das Bild als Flüssigkeit vor, in der an verschiedenen Orten unterschiedliche Konzentrationen an Farbstoff vorliegen. Der Farbstoff kann innerhalb der Flüssigkeit verlaufen. Der Transport des Farbstoffes wird durch seinen Fluss $\vec{j}$ beschrieben. Er gibt an, wie viel Farbstoff (Grauwert) pro Zeiteinheit über die grenze zwischen zwei Pixeln fließt. Diese Fluss hängt linear vom Grauwert-(Konzentrations)-Gradienten $\vec{\nabla} g$ ab. Dabei bezeichnet g das 2D-Bild und $\vec{\nabla} = \begin{pmatrix} \partial/\partial x \\ \partial/\partial y \end{pmatrix}$ den räumlichen Gradienten:

$$\vec{j}(x,y) = -D \cdot \vec{\nabla} g(x,y) \tag{8.2.1}$$

Diese Differentialgleichung wird als **1. Fick'sches Gesetz** bezeichnet. Zusätzlich gilt noch die sog. **Kontinuitätsgleichung**:

$$\frac{\partial g}{\partial t} = -\operatorname{div} \vec{j} = -\vec{\nabla} \cdot \vec{j} \tag{8.2.2}$$

Sie besagt, dass die Gesamtmenge des Farbstoffes (Grauwertes) erhalten bleiben muss. Man kann ja schließlich keinen Farbstoff vernichten. $\frac{\partial g}{\partial t}$ gibt die zeitliche Änderung der Konzentration an und $\operatorname{div} \vec{j} = \frac{\partial j_x}{\partial x} + \frac{\partial j_y}{\partial y}$ ist die Quellenstärke des Flusses. Dabei geht man davon aus, dass das Vektorfeld $\vec{j}$ den Fluss (im Prinzip die Geschwindigkeit) einer Flüssigkeit beschreibt. Die Divergenz charakterisiert dabei Stellen im Bild, an denen Farbstoff ab- oder zufließt. Vergleicht man die Situation mit einem Waschbecken, so wäre der Wasserhahn eine Quelle und der Abfluss eine Senke für den Wasserfluss. (8.2.2) besagt dann, dass die zeitliche Änderung des Grauwerte gleich dem Abfluss/Zufluss an einem bestimmten Punkt auf dem Bild sein muss. Wäre diese Gleichung nicht erfüllt, so würde an einem bestimmten Ort Farbstoff neu erzeugt bzw. vernichtet, da sich der Grauwert stärker ändert, als es durch den Zu-/Abfluss zu erklären wäre.

Setzt man (8.2.1) in (8.2.2) ein, so erhält man die **Diffusionsgleichung** oder das **2. Fick'sche Gesetz**:

$$\frac{\partial g}{\partial t} = \operatorname{div}\big(D \cdot \vec{\nabla} g\big) = D \cdot \Delta g \qquad \text{mit} \quad \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad \text{und} \quad D = \text{const} \tag{8.2.3}$$

Der letzte Schritt ist nur erlaubt, wenn die Diffusionskonstante D nicht vom Ort $(x,y)^t$ abhängt. Die partielle Differentialgleichung (8.2.3) lässt sich im Fourier-Raum relativ leicht lösen. Dazu macht man auf beiden Seiten eine Fourier-Transformation der Ortskoordinaten und erhält mit den Sätzen über Fourier-Trafos aus Abschnitt 3.6.1:

$$\frac{\partial \hat{g}(\vec{k})}{\partial t} = -4\pi^2 D \left|\vec{k}\right|^2 \hat{g}(\vec{k}) \tag{8.2.4}$$

Dies Differentialgleichung gat die Lösung:

$$\hat{g}(\vec{k},t) = \hat{g}(\vec{k},0) \cdot \exp\big(-4\pi^2 D \left|\vec{k}\right|^2 t\big) \tag{8.2.5}$$

Dieses Ergebnis kann man nun zurück transformieren. Aus der Multiplikation wird eine Faltung und die Gauß-Funktion bleibt eine Gauß-Funktion. Es ergibt sich:

$$g(\vec{x},t) = \frac{1}{[2\pi\sigma^2(t)]^{W/2}} \exp\left(-\frac{|\vec{x}|^2}{2\sigma^2(t)}\right) \circledast g(\vec{x},0) \tag{8.2.6}$$

Dabei ist $\sigma(t) = \sqrt{2Dt}$ die zeitabhängige Standardabweichung (Breite) der Gauß-Kurve. Ein Diffusionsprozess bedeutet also nichts anderes als die Faltung des Eingabebildes mit einem Gauß-Filter. Je länger die Diffusion andauert, desto breiter wird der Gauß-Filter und desto stärker wird das Ausgangsbild

$g(\vec{x}, 0)$ geglättet. Man kann also die zeit als Skalenvariable ansehen. Man setzt deswegen die Skalenvariable ξ:

$$\xi = 2Dt = \sigma^2(t)$$

Je größer also ξ ist, desto stärker ist die Glättung und desto weniger feine Strukturen enthält das Bild noch. Die folgende Abb. 8.3 zeigt wie sich ein 1D-Signal durch Diffusion verändert und wie dabei zunehmend Details verschwinden. Die kleinen Schwingungen in der periodischen Struktur sind relativ schnell verwaschen. Die groben Strukturen bleiben länger bestehen. Im Bild ganz rechts ändert sich der Skalenparameter schneller, sodass auch die groben Strukturen verwaschen und sich zum Schluss ein einheitlicher Grauwert ergibt (stabiles Bild).

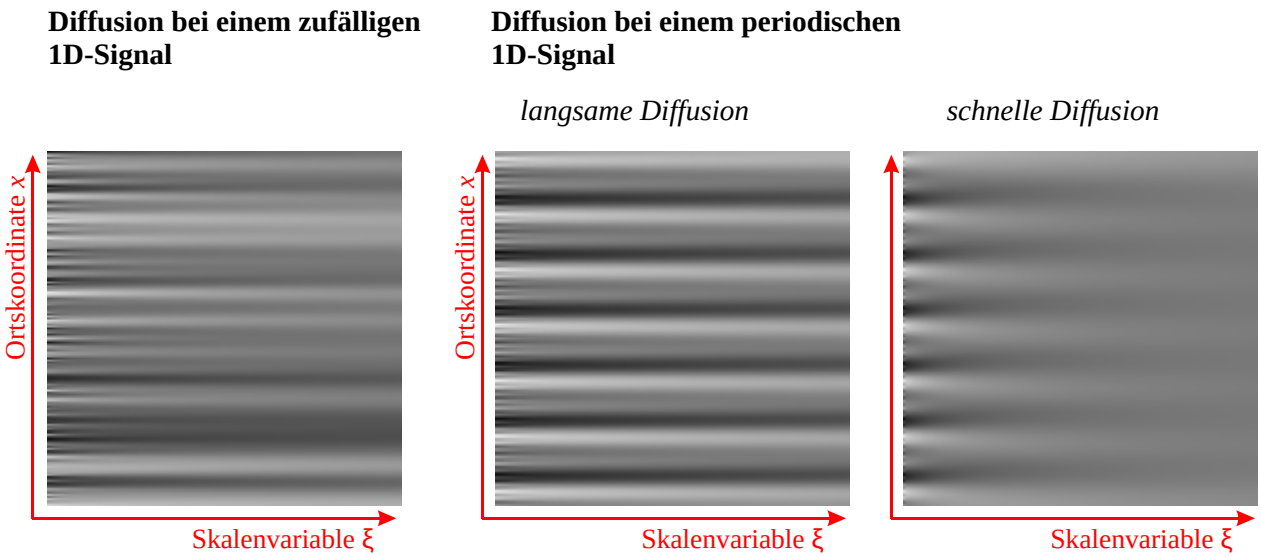


Abb. 8.3: Erzeugung eines Skalenraumes durch Diffusion am Beispiel eines zufälligen und eines perioschen 1D-Signals

8.2.2 Eigenschaften des Skalenraumes

Satz 8.1 (Minimum-Maximum-Prinzip) *Der Filter zur Erzeugung eines Skalenraumes muss dem Minimum-Maximum-Prinzip gehorchen. Es besagt, dass sich die Grauwerte an Maxima des Signals nicht erhöhen und an Minima des Signals nicht erniedrigen dürfen. Dies bedeutet, dass mit steigendem Skalenparameter ξ Information im Bild verlorengeht.*

Für den physikalischen Vorgang der Diffusion ist dies einsichtig, weil er dem Ausgleich der Konzentrationen zustrebt, als Konzentrationsunterschiede verkleinert.

Andere Filterkerne: Es zeigt sich, dass der Gauß-Filter der einzige mögliche Filter für die Erzeugung von Skalenräumen ist, da alle anderen denkbaren Filter zusätzliche Strukturen erzeugen würden.

andere Skalenräume: Man kann den Parameter D (Diffusionskonstante) zeitabhängig machen und erhält so eine schnellere oder langsamere Diffusion in Abhängigkeit von ξ. Bisher gilt $\xi = 2Dt$. Dabei steigt aber die Breite des Filters nur mit $\sigma(t) \propto \sqrt{t} \propto \sqrt{\xi}$. Die Stärke der Glättung hängt also nicht-linear von der Skala ξ ab. Durch die Setzung $D \equiv D_0 t$ erhält man eine lineare Abhängigkeit der Filterstärke von ξ **(quadratischer Skalenraum)**. Mit $D \equiv D_0\, \mathrm{e}^{t/\tau}$ sogar einen exponentiellen Zusammenhang. Die Stärke der Glättung steigt dann exponentiell mit ξ **(exponentieller Skalenraum)**.

Teil III

Einfache Merkmalsextraktion

9 Mittelung

Mittelungsfilter dämpfen schnelle Grauwertänderungen und Rauschen. Es handelt sich um Tiefpassfilter, die hohe Frequenzanteile herausfiltern. Dies führt dazu, dass Kanten nicht scharf bleiben, sondern verschmiert werden. Glättungsfilter z.B. können verwendet werden, um leichte Unebenheiten in Objekten auszugleichen, die sich durch einen konstanten Grauwert auszeichnen.

9.1 Eigenschaften

Verschiebungsfreiheit: Ein Glättungsfilter darf die Position eines Objekte (einer Kante) nicht verändern, da solche Fehler später nicht mehr ausgeglichen werden können. Eine Verschiebung eines Objektes bedeutet die Multiplikation des Fourier-Transformierten mit einem Phasenfaktor. Daraus kann man folgern, dass die Transferfunktion (FT der Punktantwort) eines Mittelunsgfilters reell sein muss (man sagt auch *nullphasig*). Nach den Eigenschaften der Fouriertransformation (siehe Abschnitt 3.6) ist damit die Punktantwort (Filtermaske) eines Mittelungsfilters symmetrisch.

Satz 9.1 (Verschiebungsfreiheit) *Die Punktantwort eines Mittelungsfilters muss symmetrisch sein. Damit ist die Transferfunktion reell und der Filter ist nullphasig. Es gilt:*

$$\begin{aligned} 1D: &\quad h_{-n} = h_n \quad \Leftrightarrow \quad \hat{h}_\nu \in \mathbb{R} \\ 2D: &\quad h_{m,-n} = h_{m,n}, \quad h_{-m,n} = h_{m,n} \quad \Leftrightarrow \quad \hat{h}_{\nu,\mu} \in \mathbb{R} \end{aligned}$$

Damit ergibt sich aus Gleichung (7.1.1) aus Abschnitt 7.1.2 die folgende Darstellung für die Transferfunktion eines Mittelungsfilters

$$\hat{h}(k) = h_0 + 2 \cdot \sum_{\nu=1}^{R} h_\nu \cos(\nu \pi k)$$

Über eine ähnliche Rechnung, wie in 7.1.2 erhält man auch eine Formel für die 2D-Transferfunktion:

$$\begin{aligned} \hat{h}(\vec{k}) = h_{00} & \\ & + 2 \sum_{\nu=1}^{R} h_{0,\nu} \cos(\nu \pi k_x) + 2 \sum_{\nu=1}^{R} h_{\nu,0} \cos(\nu \pi k_y) \\ & + 4 \sum_{\nu=1}^{R} \sum_{\mu=1}^{R} h_{\nu,\mu} \cos(\nu \pi k_x) \cdot \cos(\mu \pi k_y) \end{aligned}$$

Auch hier sind nur (symmetrische) Cosinus-Komponenten enthalten.

Satz 9.2 (Erhaltung des Mittelwertes) *Der Mittelwert soll bei einer Glättung erhalten bleiben. Deswegen müssen sich die Filterkoeffizienten zu 1 addieren:*

$$1D: \quad \sum_n h_n = 1$$

$$2D: \quad \sum_m \sum_n h_{m,n} = 1$$

Satz 9.3 (Monoton fallende Transferfunktion) *Die Transferfunktion eines Mittelunsgfilters soll mit steigender Wellenzahl monoton gegen 0 fallen:*

$$\hat{h}(k_2) \leq \hat{h}(k_1) \quad \textit{für} \quad k_2 > k_1$$

Dies bedeutet, dass größere Wellenzahlen stärker gedämpft werden, als kleine Wellenzahlen. Damit werden kleine Strukturen stärker unterdrückt, als grobe Strukturen. Mittelungsfilter sind also Tiefpassfilter.

Satz 9.4 (Isotropie) *Die Glättung soll oft unabhängig von der Richtung sein. Dies bedeutet, dass sowohl die Punktantwort (Filtermaske), als auch die Transferfunktion isotrop sein sollen. Sie hängen also nur vom Betrag des Abstandes bzw. der Wellenzahl ab, nicht von der Richtung:*

$$h(\vec{x}) \equiv h(|\vec{x}|) \quad \textit{und} \quad \hat{h}(\vec{k}) \equiv \hat{h}(|\vec{k}|)$$

9.2 Rechteckfilter

Der einfachste Mittelungsfilter ist ein Rechteckfilter. Seine Filtermaske hat auf allen Plätzen den Koeffizienten 1. Der Filter ist natürlich auf 1 normiert:

$$\text{1D:} \ \mathcal{R}_{1D}^{(3)} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} \qquad \text{2D:} \ \mathcal{R}_{2D}^{(9)} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Dieser Filter ist auch als laufender Mittelwertfilter bekannt. Die folgende Abbildung zeigt Beispiele für die Anwendung des Filters

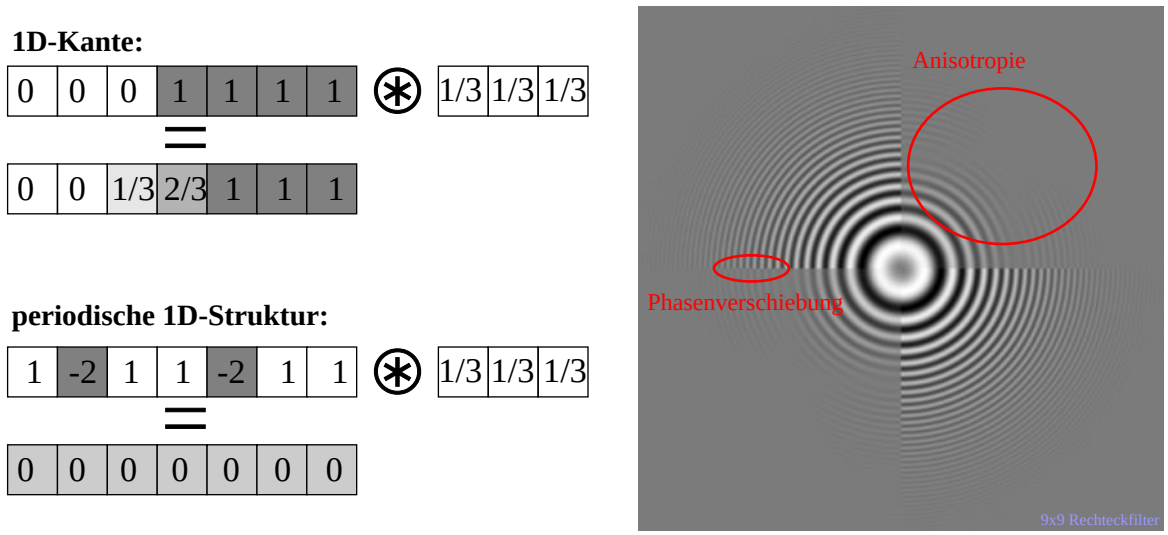


Abb. 9.1: Anwendung von 1D- und 2D-Rechteckfiltern. Im 2D-beispiel wurde der Filter rechts oben und links unten angewendet. Der Rest des Bildes zeigt das Ausgangsmuster.

Man sieht in Abb. 9.1 deutlich, dass die Position einer 1D-Kante erhalten bleibt. Die gezeigte periodische Struktur mit der Periode 3 Pixel verschwindet ganz bei Anwendung des Filters. Der 2D-Filter arbeitet nicht-isotrop. In bestimmten Richtungen wird stärker geglättet als in anderen. Die Anisotropie des Filters tritt besonders in seiner Transferfunktion zu Tage. Sie ist in Abb. 9.2 gezeigt.

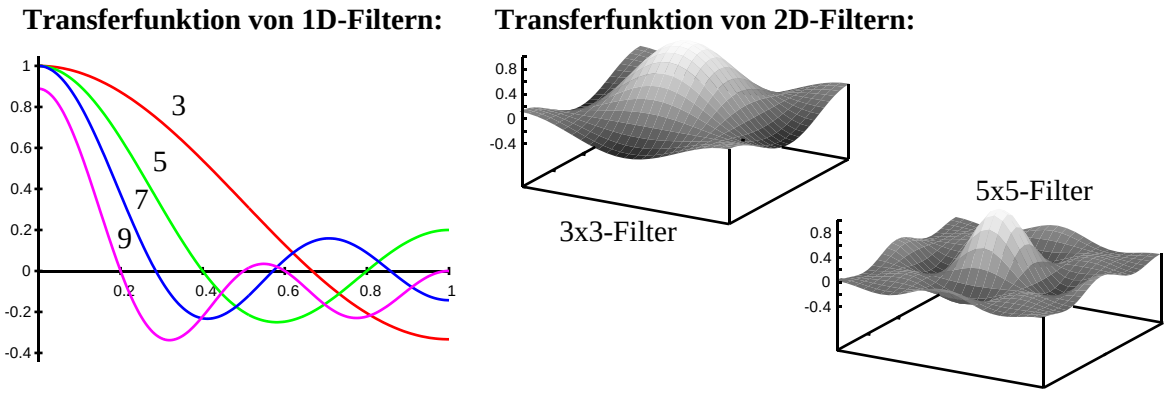
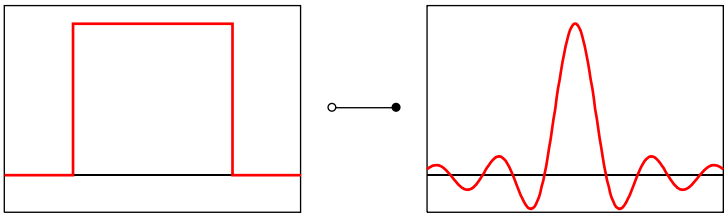


Abb. 9.2: Transferfunktionen von 1D- und 2D-Rechteckfilter verschiedener Breite

Man sieht, dass die Transferfunktionen nicht monoton gegen 0 fallen. Die Transferfunktion eines idealen Rechteckfilters ist (wie schon öfter besprochen die sinc-Funktion:



Sie nähert sich für große k zwar der 0, oszilliert aber um diese und fällt wie $1/k$ (also recht langsam) gegen 0 ab. Die Stellen an denen die Transferfunktio kleiner als 0 wird entsprechen einer Phasenverschiebung um $180°$ (Vorfaktor -1). Damit ist das Rechteckfilter weder isotrop (ab 2D), noch fällt es monoton, noch ist es nullphasig. Das Rechteckfilter ist also kein gutes Glättungsfilter.

Es bleibt noch anzumerken, dass das 2D-Rechteckfilter separabel ist. Es lässt sich als Faltung zweier 1D-Filter darstellen und kann so schnell berechnet werden:

$$\mathcal{R} = \mathcal{R}_x \circledast \mathcal{R}_y \quad \Leftrightarrow \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} \circledast \frac{1}{3}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}$$

9.3 Binomialfilter

Die einfachste 1D-Glättungsfiltermaske ist $\vec{B} = \frac{1}{2}[1 \quad 1]$. Sie mittelt zwei benachbarte Bildpunkte. Das Ergebnisbild liegt (weil die breite eine gerade Zahl ist) auf einem Zwischengitter. Man kann diesen Filter mehrfach (R mal) hintereinander anwenden und erhält dadurch:

$$\frac{1}{2^R}\underbrace{[1 \quad 1] \circledast [1 \quad 1] \circledast \ldots \circledast [1 \quad 1]}_{R \text{ mal}}, \qquad \mathcal{B}^R = \mathcal{B}\mathcal{B}\ldots\mathcal{B}$$

Dabei ergeben sich Filter mit den Koeffizienten, die den Binomialkoeffizienten entsprechen:

R	Vorfaktor	Filtermaske
1	1/2	1 1
2	1/4	1 2 1
3	1/8	1 3 3 1
4	1/16	1 4 6 4 11
5	1/32	1 5 10 10 5 1
6	1/64	1 6 15 20 15 6 1

Tabelle 9.1: Binomialfiltermasken

Von diesen Filtern sind natürlich vor Allem diejenigen mit ungerader Breite interessant.

Breite der Faltungsmaske, Binomialverteilung: Die Koeffizienten der Faltungsmaske entsprechen der Binomialverteilung

$$f_n = \binom{N}{n} \cdot p^n \cdot (1-p)^{N-n}, \qquad 0 \le n < N$$

Dabei ist N die Gesamtzahl der Experimente und p die Wahrscheinlichkeit für einen Treffer. f_n gibt dann die Wahrscheinlichkeit an genau n Treffer zu erhalten. Für Mittelwert und Varianz gilt:

$$\mu = Np, \qquad \sigma^2 = Np(1-p)$$

Für die Binomialfilter möchte man eine um $N/2$ zentrierte Verteilung, wählt also $p = \frac{1}{2}$. Damit ist die Standardabweichung eine Binomialfilters

$$\sigma = \sqrt{\frac{R}{4}} \propto \sqrt{R}$$

und steigt nur mit der Wurzel von R. Die Glättungswirkung steigt also bei größeren Filtermasken immer langsamer an.

Es zeigt sich, dass die Binomialverteilung für große N in die Gauß-Verteilung übergeht. Ein Binomialfilter ist also die natürliche Entsprechung eines Gauß-Filters für begrenzte Faltungsmasken. Die folgende Abb. 9.3 zeigt die Binomialverteilung.

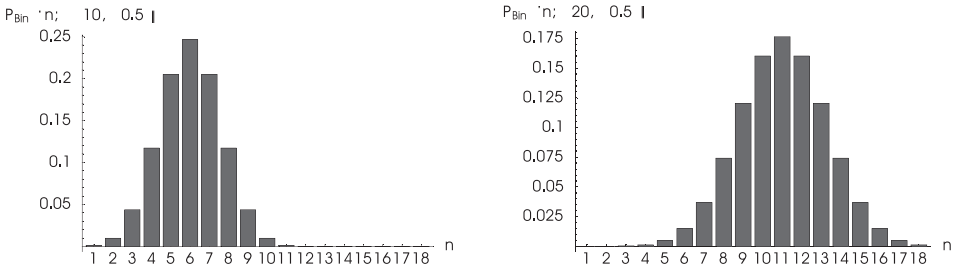


Abb. 9.3: Binomialverteilung für $p = \frac{1}{2}$

2D-Filter: Die 2D-Filter ergeben sich durch Faltung zweier 1D-Filter:

$$\mathcal{B} = \mathcal{B}_x \circledast \mathcal{B}_y \qquad \Leftrightarrow \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \circledast \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Transferfunktion: Da sich die Filtermaske durch Faltung im Ortsraum berechnet kann man die Transferfunktion einfach durch Multiplikation im Fourierraum berechnen. Es gilt:

$$B^{(R)}(k) = \cos^R(\pi k/2).$$

Dabei wurde verwendet, dass sich die Transferfunktion des einfachsten Filters $\vec{B} = \frac{1}{2}[1 \quad 1]$ zu $B(k) = \cos(\pi k/2)$ ergibt. Die folgende Abb. 9.4 zeigt diese Transferfunktion.

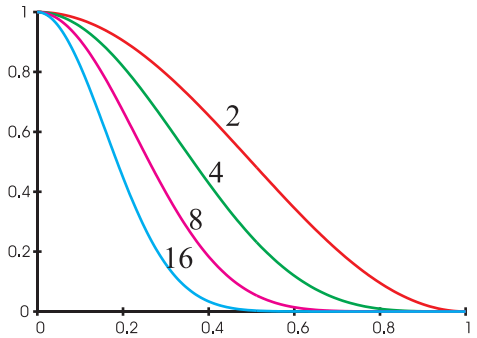


Abb. 9.4: Transferfunktion verschiedener Binomialfilter

Man sieht, das die Transferfunktion monoton gegen 0 fällt. Es handelt sich hier also um einen wesentlich besseren Glättungsfilter, als der Rechteckfilter. Der Filter ist auch tatsächlich nullphasig. Es kommt also zu keiner Phasenverschiebung.

Die Transferfunktion des 2D-Filter hat eine ähnliche Struktur:

$$b^{(R)}(\vec{k}) = \cos^R(\pi k_x/2) \cdot \cos^R(\pi k_y/2).$$

Diese Transferfunktion ist nahezu (aber nicht ganz) isotrop. Eine leichte Rest-Anisotropie bleibt übrig. Bei einer isotropen Transferfunktion würde nur ein $\cos$-Term auftreten, in dem $|\vec{k}|$ auftaucht.

Anwendung: Wo im idealen, kontinuierlichen Fall ein Gauß-Filter Verwendung findet wird im diskreten Fall ein Binomialfilter angewendet. Ein Binomialfilter eignet sich besonders um (mittelwertfreies) Gauß'sches Rauschen zu entfernen. Bei Salt-and-Pepper-Rauschen ist er weniger gut, da er die Fehler über die Umgebung des Rauschpeaks nur verteilt. Um solches Rauschen zu Filtern sind Medianfilter besser geeignet. Bei der Rauschunterdrückung geht natürlich Detailauflösung verloren, da ja nicht nur das Rauschen, sondern auch das Bild geglättet wird. hier ist also Vorsicht geboten, wenn sich das Rauschen auf der gleichen Skala verändert, wie das eigentliche Bild.

Schnelle Berechnung: Es ist günstiger die Berechnung durch wiederholtes Anwenden der kleinen $[1 \quad 1]$-Masken auszuführen, anstatt mit der großen Filtermaske zu falten. Die hier auftretenden Multiplikationen sind nur $\div 2$ und $*2$, sodass sie durch Bit-Shift-Operationen ausgeführt werden können. Diese beherrschen alle gängigen CPUs und sie sind viel schneller als tatsächliche Multiplikationen.

Die Anwendung einer $(R+1) \times (R+1)$-Filtermaske benötigt bei naiver Implementierung $\mathcal{O}(R^2)$ Multiplikationen und Additionen pro Pixel. Bei der Zerlegung der Maske in $[1 \quad 1]$-Masken und Anwendung in zwei Richtungen benötigt man aber nurmehr $\mathcal{O}(R)$ Additionen. Die Multiplikationen können, wie oben beschrieben vereinfacht werden.

9.4 Schnelle großräumige Mittelung

Im vorherigen Abschnitt wurde erwähnt dass die Standardabweichung (und damit die Glättungswirkung) eines Binomialfilters nur mit $\sqrt{R}$ wächst (siehe S. 59). Um eine starke, großräumige Glättung zu erreichen kann man zwar die Filtermaske immer weiter vergrößern. Der zusätzliche Glättungseffekt einer solchen Vergrößerung wird aber immer kleiner, sodass man andere Verfahren anwenden muss.

9.4.1 Mehrschrittmittelung

Eine Idee ist es zunächst eine normale Binomial-Glättung durchzuführen und danach eine weitere Glättung, die aber mit einer größeren Schrittweite als die $[1 \quad 1]$-Maske ($\vec{B}_x$) arbeitet usw. Eine mögliche Maske für den zweiten Mittelungsschritt ist:

$$\vec{B}_{2x} = \frac{1}{4}\begin{bmatrix}1 & 0 & 2 & 0 & 1\end{bmatrix}$$

Eine solche Maske tastet nur mehr jeden zweiten Bildpunkt ab. Analog lassen sich noch größere Masken $\vec{B}_{4x}, \vec{B}_{8x}$ usw. erzeugen. Man kann dann diese Masken so zu einem Mehrschritt-Mittelungsoperator zusammensetzen:

$$\mathcal{B}_k := \underbrace{\mathcal{B}^{(R)}_{2^{S-1}x} \cdots \mathcal{B}^{(R)}_{8x}\mathcal{B}^{(R)}_{4x}\mathcal{B}^{(R)}_{2x}\mathcal{B}^{(R)}_{x}}_{S \text{ mal}}$$

Dabei gibt der Parameter R die Größe der Anfangsmaske an. $R = 2$ entspricht $\vec{B}^{(2)} = \frac{1}{2}[1 \;\; 2 \;\; 1]$. $R = 4$ entspricht $\vec{B}^{(4)} = \frac{1}{2}[1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]$. Die Zahl im unteren Index gibt dann die Spreizung des Gitters an.

Effiziente Implementierung, Mehrgittermittelung: Die jeweils nächsten Mittelungsoperatoren benutzen nicht alle Punkte, die der vorherige Operator erzeugt. So tastet $\mathcal{B}^{(R)}_{2x}$ nur jeden zweiten Punkt ab, obwohl $\mathcal{B}^{(R)}_{x}$ auch die dazwischen liegenden Punkte erzeugt. Man kann somit die Operatoren so modifizieren, dass sie nur noch jeden zweiten Punkt ausgeben. Dies erreicht man, indem die Maske nicht um einen Bildpunkt, sondern um zwei Bildpunkte vorgerückt wird. Man kennzeichnet einen solchen Operator durch einen speziellen Index: $\mathcal{B}_{x|2}$ mittelt etwa in x-Richtung und wird jeweils um 2 Positionen weitergeschoben. Man kann dann obige kaskadierte Glättung $\mathcal{B}_k$ auch so schreiben:

$$\mathcal{B}_k := \underbrace{\mathcal{B}^{(R)}_{2|x} \cdots \mathcal{B}^{(R)}_{2|x}\mathcal{B}^{(R)}_{2|x}}_{S \text{ mal}}$$

Dies bedeutet, dass die späteren Optimierungen auf immer kleineren Gittern arbeiten, also wesentlich schneller werden.

9.4.2 Rekursive Mittelung

Ein rekursives Filter (siehe Abschnitt 7.3) hat eine unendliche Punktantwort, die von der Maskengröße unabhängig ist. Man kann deswegen Glättungsfilter erstellen, deren Glättungsgrad über einen Parameter einstellbar ist, ohne dabei die Maskengröße zu verändern. Somit sind mit gleichbleibender Komplexität sehr starke und sehr schwache Mittelung möglich. Ein Beispiel für ein parametrisiertes Tiefpassfilter ist etwa:

$$^{\pm}\mathcal{A}: \quad g'_n = \cdot g'_{n\pm1} + \alpha \cdot \big(g_n - g_{n\mp1}\big), \quad 0 < \alpha \leq 1$$

Dabei handelt es sich allerdings um ein kausales Filter, das in eine bestimme Richtung läuft. Solche Filter erfüllen nicht die Verschiebungsreiheit. Nimmt man aber an, dass man ein vor- und ein rückwärtslaufendes kausales Filter ${}^{\pm}\mathcal{A}$ hat, so kann man diese zu einem verschiebungsfreien Filter kombinieren. Die Transferfunktionen der obigen Filter sind:

$$ {}^{\pm}\hat{A}(k) = a(k) \pm \mathrm{i} \cdot b(k) $$

kombiniert man diese Transferfunktionen so, das sich eine reelle Transferfunktion ergibt, so erhält man ein verschiebungsfreies Mittelungsfilter. Dazu gibt es zwei Möglichkeiten (im Fourier-Raum):

$$ \begin{array}{ll} \text{Addition} & \hat{A} = \frac{1}{2}\left[{}^{+}\hat{A} + {}^{-}\hat{A}\right] = a(k) \\ \text{Multiplikation} & \hat{A} = {}^{+}\hat{A} \cdot {}^{-}\hat{A}\big] = a^2(k) + b^2(k) \end{array} $$

Die Multiplikation entspricht im Realraum einer Hintereinanderausführung, sodass man mit obigem Filter leicht auch ein 2D-Filter erhält:

$$ \mathcal{A}_{2D} = \mathcal{A}_x \mathcal{A}_y = {}^{+}\mathcal{A}_x \, {}^{-}\mathcal{A}_x \, {}^{+}\mathcal{A}_y \, {}^{-}\mathcal{A}_y $$

Wie bereits gesagt sind die Vorteile eines solchen Filters die Unabhängigkeit des Rechenaufwandes vom Glättungsgrad und die leichte Einstellbarkeit der Glättung. Die 2D-Filter haben aber eine große Anisotropie, sodass man evtl. zusätzliche Filter in Diagonalrichtung einsetzen muss. Ist $\alpha = 2^{-l}, \quad l \in \mathbb{N}$, so kann man den Filter sogar ganz ohne Multiplikation und nur mit Bit-Shifts berechnen.

9.5 nichtlineare und steuerbare Mittelung

9.5.1 Medianfilter

Die Grundeigenschaften von Medianfiltern sind im Abschnitt 7.2 über Rangordnungfilter bereits ausführlich behandelt worden. Dort wurde auch erklärt, wie ein solches Filter besonders Salt-and-Pepper- bzw. Impuls-Rauschen unterdrückt (siehe Abb. 7.3). Die Strukturen im Bild (wie Kanten oder Rampen) bleiben dabei erhalten. Medianfilter sind bei gauß'schem Rauschen eher schlecht geeignet.

9.5.2 gewichtete und einstellbare Mittelung

Oft sind zu Messdaten die Fehler bekannt (in Form der Varianz). Bei Bildern hat man also neben dem eigentlichen Bild g_{mn} noch ein Varianzbild σ^2_{mn}. Möchte man nun über alle Punkte mitteln und dabei das statistische Gewicht eines Punktes nach seinem Fehler richten, so wird jeder Punkt mit dem Kehrwert seiner Varianz gewichtet:

$$ \overline{g} = \text{const} \cdot \sum_{m,n} \frac{1}{\sigma^2_{mn}} \cdot g_{mn} $$

Oft betrachtet man deswegen auch statt dem Varianzbild ein Bild der Wichtungsfaktoren

$$ w_{mn} = \frac{1}{\sigma^2_{mn}}. $$

Dieses Konzept kann auf eine Faltung erweitert werden:

Definition 9.1 (normalisierte Faltung) *Sei h eine Faltungsmaske, g ein beliebiges Bild und w das zugehörige Wichtungsbild. Dann ist die normalisierte Faltung definiert als:*

$$g' = \frac{h \circledast (w \cdot g)}{h \circledast w}$$
$$w' = h \circledast w$$

Die $\cdot$-Operation entspricht einer pixelweisen Multiplikation.

Man kann diese Art der Faltung z.B. nutzen, indem man die Kantenstärke als Wichtungsbild verwendet und so Kanten weniger glättet als den Rest des Bildes.

Neben der normalisierten Faltung kann man auch Filter designen, deren Mittelunsgrad über einen Parameter α anpassbar ist. Macht man diesen Parameter von der aktuellen Umgebung im Bild abhängig, so erhält man ebenfalls eine einstellbare/gewichtete Mittelung.

10 Kantendetektion

Kantendetektion

10.1 differentielle Merkmalsbeschreibung

Hier soll kurz aufgezählt werden, wie man Bildmerkmale (Kanten, Ecken ...) mit ersten und zweiten Ableitungen beschreibt.

Kante an einer Kante findet man eine schnelle Änderung im Signal (große erste Ableitung) senkrecht zur Kantenrichtung

Ecke Bei einer Ecke handelt es sich um eine Kante, die zusätzlich eine große Krümmung senkrecht zur Gradientenrichtung aufweist

Linie hier ist die Krümmung senkrecht zur Linie groß. Es handelt sich um ein lokales Extremum, also verschwindet die 1. Ableitung.

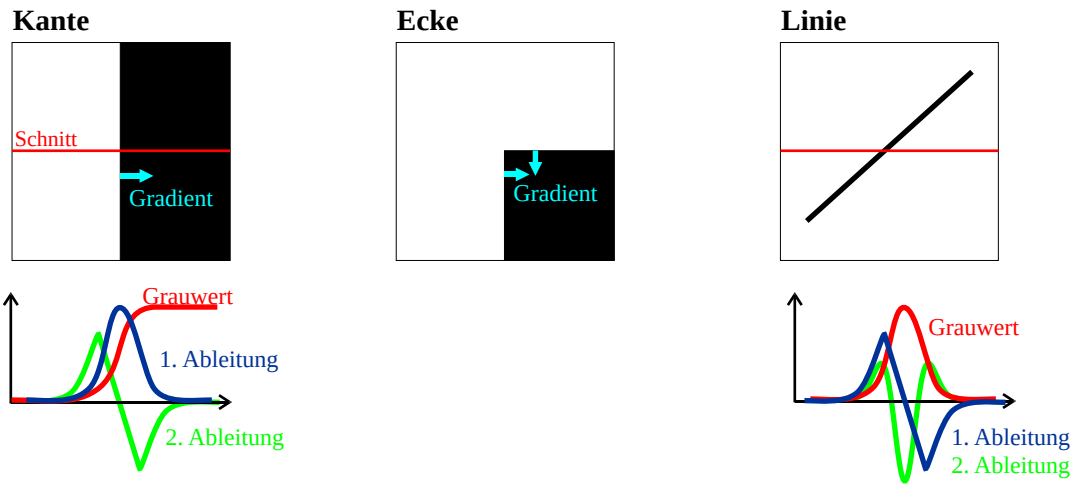


Abb. 10.1: differentielle Merkmalsbeschreibung

10.2 Ableitungsoperatoren

Aus der Einleitung kann man schließen, dass man sich hier im wesentlichen mit den ersten und zweiten Ableitungen beschäftigt. Um passende Filtermasken zu erstellen ist es auch günstig sich die Fourier-Transformierten der entsprechenden Operatoren anzusehen. In Abschnitt 3.6 wurde folgender Satz erwähnt:

$$\frac{\partial g}{\partial x} \quad \circ\!\!-\!\!\bullet \quad 2\pi \mathrm{i} \cdot k \cdot \hat{g}(k)$$

Die Ableitung $\frac{\partial g}{\partial x}$ lässt sich auch als Anwendung des Operators $\frac{\partial}{\partial x}$ auf das Bild g sehen. Daraus erhält man folgende Beziehungen:

$$\frac{\partial}{\partial x} \quad \circ\!\!-\!\!\bullet \quad 2\pi \mathrm{i} k \tag{10.2.1}$$

$$\frac{\partial^2}{\partial x^2} \quad \circ\!\!-\!\!\bullet \quad -4\pi^2 k^2 \tag{10.2.2}$$

$$\vec{\nabla} \quad \circ\!\!-\!\!\bullet \quad 2\pi \mathrm{i} \vec{k} \tag{10.2.3}$$

$$\Delta = \vec{\nabla}^2 x \quad \circ\!\!-\!\!\bullet \quad -4\pi^2 \left|\vec{k}\right|^2 \tag{10.2.4}$$

In den letzten beiden Gleichungen wurden der Gradienten- und der Laplace-Operator benutzt:

$$\vec{\nabla} = \begin{pmatrix} \partial/\partial x_1 \\ \partial/\partial x_2 \\ \vdots \end{pmatrix} \qquad \Delta = \vec{\nabla}^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \dots$$

10.3 Eigenschaften

Symmetrie/Verschiebungsfreiheit: Ein Ableitungsfilter darf die Position eines Objekte (einer Kante) nicht verändern, da solche Fehler später nicht mehr ausgeglichen werden können. Wie man bereits an den Fourier-Transformieren idealen Operatoren sieht, sollte die Transferfunktion einer *ersten Ableitung* komplex sein. Daraus ergibt sich mit den Eigenschaften der Fourier-Transformation, dass die Filtermaske komplex und antisymmetrisch sein muss:

$$h_{-n} = -h_n$$

Hat die Maske eine ungeradzahlige Breite, so bedeutet diese Bedingung, dass das mittlere Element 0 ist.

Betrachtet man die *zweite Ableitung*, so müssen die Masken symmetrisch und reell sein.

Phasenverschiebung: Es zeigt sich, dass Ableitungsfilter eine definierte Phasenverschiebung auf ein Signal geben. Dazu betrachtet man die erste und zweite Ableitung einer Sinus-Funktion:

$$\underbrace{\frac{\partial}{\partial x} \sin x = \cos x}_{90^\circ \text{ Phasenverschiebung gegenüber } \sin x} \qquad \underbrace{\frac{\partial^2}{\partial x^2} \sin x = -\sin x}_{180^\circ \text{ Phasenverschiebung gegenüber } \sin x}$$

Dies bedeutet auch, dass eine erste Ableitungen Extrema auf Nulldurchgänge und große Steigungen auf Extrema abbildet. Die zweite Ableitung führt Extrema in Extrema über.

Satz 10.1 (Unterdrückung des Mittelwertes) *Der Mittelwert soll bei allen Ableitungen unterdrückt werden. Ein solches Filter muss also konstante Anteile des Signals entfernen. Dies lässt sich erreichen, wenn sich die Filterkoeffizienten zu null addieren:*

$$\begin{aligned} 1D: &\quad \sum_n h_n = 0 \\ 2D: &\quad \sum_m \sum_n h_{m,n} = 0 \end{aligned}$$

Diese Bedingung lässt sich auch für die Transferfunktion schreiben. Im Fourier-Raum wird der konstante Anteil des Signals auf das zentrale Pixel abgebildet ($k = 0$). Das zentrale Pixel der Transferfunktion des Filters muss also null sein:

$$\hat{h}(0) = 0$$

Isotropie: Ein idealer Kantendetektor sollte Kanten in beliebigen Richtungen mit der gleichen Empfindlichkeit erkennen.

Transferfunktion:

- *1. Ableitung:* Die Transferfunktion muss imaginär und antisymmetrisch sein (siehe oben). Nach Gleichung (7.1.2) aus Abschnitt 7.1.2 hat man dann folgende Darstellung der 1D-Transferfunktion:

$$\hat{h}(k) = -2\mathrm{i} \cdot \sum_{n'=1}^{R} h_{n'} \sin\left(n'\pi k\right)$$

  Dabei ist R die Breite der Filtermaske h_n.

- *2. Ableitung:* Die Transferfunktion muss reell und symmetrisch sein (siehe oben). Nach Gleichung (7.1.1) aus Abschnitt 7.1.2 hat man dann folgende Darstellung der 1D-Transferfunktion:

$$\hat{h}(k) = h_0 + 2 \cdot \sum_{n'=1}^{R} h_{n'} \cos\left(n'\pi k\right)$$

10.4 Gradientenbasierte Ableitungsfilter

Ideale Transferfunktion? Zunächst sollen Filter entwickelt werden, die Kanten über den Gradienten des Bildes detektieren. Dazu stellt sich zunächst die Frage in wieweit die ideale Transferfunktion $2\pi\mathrm{i}\vec{k}$ von einem Filter erfüllt werden soll. Es zeigt sich, dass die Bedingung diese Funktion exakt zu erfüllen zu stark ist, da z.B. auch nach einer Glättung noch eine Kante detektiert werden kann. Man fordert daher, dass der Ableitungsfilter die ideale (lineare) Transferfunktion nur in einem kleinen Bereich um $k = 0$ erfüllt (nutze evtl. Taylor-Entwicklung). Die folgende Abb. 10.2 verdeutlicht das anhand der Nacheinanderausführung einer Gauß-Glättung und eines idealen Ableitungsfilters (Multiplikation der Filtermasken im Fourier-Raum, Faltung im Realraum).

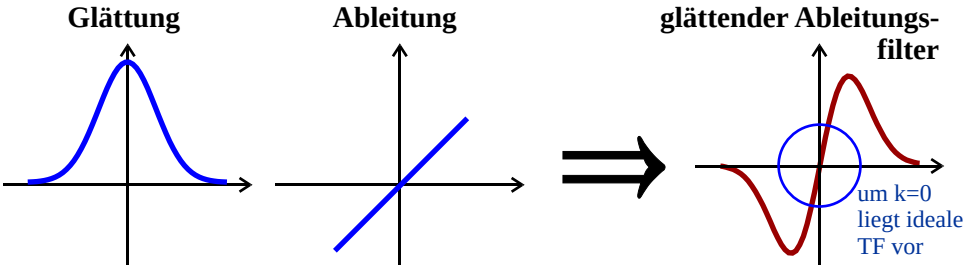


Abb. 10.2: Kombination von Glättung und Ableitung in einem Filter

Gradientenbetrag: Die ersten Ableitungen in den verschiedenen Richtungen ergeben einen vektorwertigen Filter $\vec{\mathcal{D}}$. In dem damit gefilterten Signal muss man anschließend die Minima und Maxima finden. Oft ist es sinnvoll die Richtungsinformation zu unterdrücken und man möchte nur die Kantenstärke in einem Bild erhalten. Dann muss man den Betrag des Gradienten berechnen. Dies lässt sich mit folgender Operation bewerkstelligen:

$$\vec{\mathcal{D}} = \begin{pmatrix} \mathcal{D}_x \\ \mathcal{D}_y \end{pmatrix} \qquad \Rightarrow \quad |\vec{\mathcal{D}}| = \sqrt{\mathcal{D}_x \cdot \mathcal{D}_x + \mathcal{D}_y \cdot \mathcal{D}_y}$$

Die Symbole $+$ und $\cdot$ stehen für punktweise Addition und Multiplikation. Diese Operationen können als dyadische Punktoperationen implementiert werden. Man kann dann eine LUT verwenden, um die Berechnung zu beschleunigen. Manchmal verwendet man auch eine einfachere Betragsberechnung:

$$|\vec{\mathcal{D}}| = |\mathcal{D}_x| + |\mathcal{D}_y|$$

Da hier weniger Operationen benötigt werden, ist diese Betragsbildung auch schneller. Dafür ist diese Betragsberechnung wesentlich anisotroper als die obige Variante. Dies sieht auch in der folgenden Abb. 10.3, in der die Punkte (x, y) mit $\left|\begin{pmatrix} x \\ y \end{pmatrix}\right| = 1$ aufgetragen sind. Es bleibt noch zu bemerken, dass mathematisch die erste Betragsberechnung der euklidischen Norm und die zweite Betragsberechnung der L1-Norm entspricht.

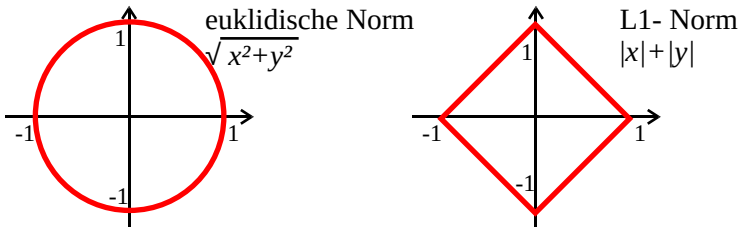


Abb. 10.3: Darstellung der Punkte mit Betrag 1 zur euklidischen und zur L1-Norm

mögliche Fehler: Bei der Gradientenberechnung kann man zwei wesentliche Fehler machen:

- falsche Richtung
- falscher Betrag/falsche Länge

diskrete Differenzen: Die einfachsten Ableitungsfilter erhält man aus der diskreten Darstellung der Ableitung:

$$\begin{aligned}
\frac{\partial g}{\partial x} &\approx \frac{g(x) - g(x - \Delta x)}{\Delta x} && \text{(Rückwärtstransformation)} \\
&= \frac{g(x + \Delta x) - g(x)}{\Delta x} && \text{(Vorwärtstransformation)} \\
&= \frac{g(x + \Delta x) - g(x - \Delta x)}{2 \cdot \Delta x} && \text{(symmetrische Differenzen)}
\end{aligned}$$

Aus diesen Gleichungen kann man einfache Filtermasken ableiten. Der Index $\bullet$ zeigt an, dass der Filter an diesem Punkt aufsetzt. Es sind dann immer noch entsprechende Masken der BReite 3 angegeben:

$$^{-}\vec{D}_x = [1_\bullet \;\; -1] = [1 \;\; -1 \;\; 0] \qquad ^{+}\vec{D}_x = [1 \;\; -1_\bullet] = [0 \;\; 1 \;\; -1] \tag{10.4.1}$$

$$\vec{D}_{2x} = [1 \;\; 0 \;\; -1] \tag{10.4.2}$$

Nur die Makse $\vec{D}_{2x}$ ist symmetrisch. Man kann $^{\pm}\vec{D}_x$ als symmetrisch auffassen, wenn man das Ergebnis an die Zwischenstelle zwischen den zwei $\neq 0$-Einträgen schreibt. Dann verschieben sich aber alle Muster im Bild um eine halbe Pixelbreite. Die Transferfunktion des Filters $\vec{D}_{2x}$ ist sehr einfach:

$$\hat{d}_{2x} = \mathrm{i} \cdot \sin(\pi k)$$

Aus der Taylor-Entwicklung des Sinus ersieht man, dass für $|k| \ll 1$ gilt $\sin(x) \approx x$, sodass diese Transferfunktion um $k = 0$ der idealen Transferfunktion entspricht. Die folgende Abb. 10.4 zeigt diese Transferfunktion.

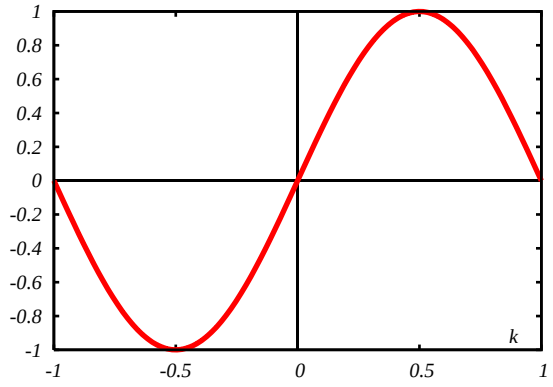


Abb. 10.4: Transferfunktion des Filters $\vec{D}_{2x}$

Es zeigt sich eine Ähnlichkeit zur schematischen Darstellung in Abb. 10.2. Der Filter sollte sich also als Kombination aus Glättung und Ableitung darstellen lassen. Mit Hilfe des einfachen Binomialfilters $\vec{B}_x = \frac{1}{2}[1\ \ 1]$ kann man $\vec{D}_{2x}$ auch über die Rückwertsdifferenzen darstellen:

$$\vec{D}_{2x} = {}^{-}\mathcal{D}_x\, \mathcal{B}_x = [1_\bullet\ \ -1] \circledast \frac{1}{2}[1\ \ 1] = \frac{1}{2}[1\ \ 0\ \ -1]$$

Es zeigt sich das solche einfachen Ableitungsfilter stark anisotrop sind und große Betragsfehler aufweisen, die zusätzlich von der Kantenorientierung abhängen.

10.5 Laplace-Filter

Man kann Kanten auch finden, wenn man die Nulldurchgänge der zweiten Ableitung eines Signals sucht. Dazu betrachte man zunächst Abb. 10.5. Es ist dabei zu beachten, das nicht jeder Nulldurchgang einer Kante entspricht. Die kleinen Nulldurchgänge werden durch Rauschen im Bild verursacht. Wichtig ist also, das vor und nach dem Nulldurchgang Maxima, bzw. Minima liegen, die über dem allgemeinen Rauschpegel des Signals liegen. Damit ist die Kantendetektion über die zweite Ableitung viel rauschanfälliger, als die Anwendung die Bestimmung über Gradienten.

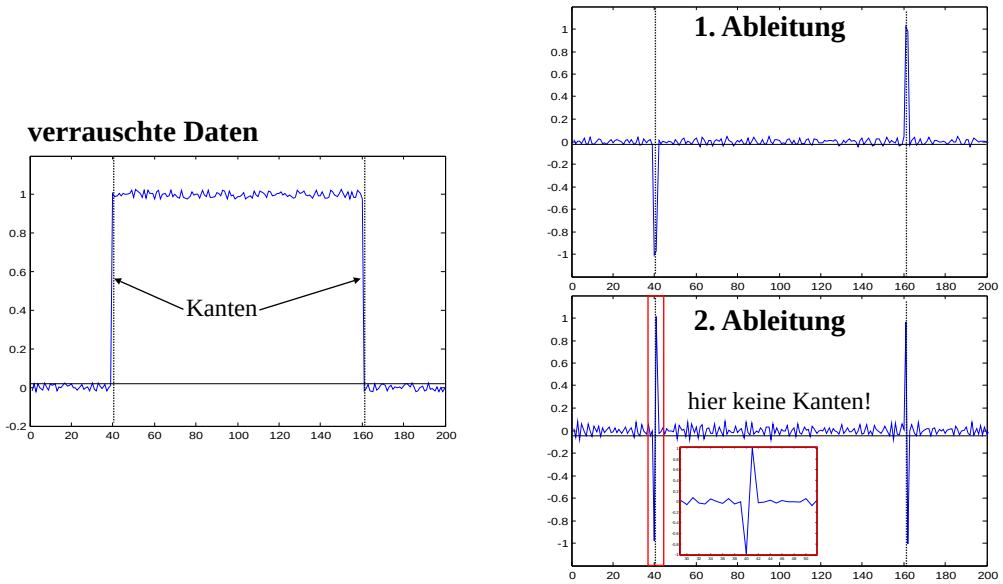


Abb. 10.5: erste und zweite Ableitung von verrauschten Daten

Laplace-Operator: Zur Berechnung der zweiten Ableitung verwendet man den Laplace-Operator $\mathcal{D}$. Seine Faltungsmaske ist:

$$\textbf{1D}: \qquad \vec{D}_x^2 = {}^-\vec{D}_x \circledast {}^+\vec{D}_x = [1_\bullet \;\; -1] \circledast [1 \;\; -1_\bullet] = [-1 \;\; 2 \;\; -2]$$

$$\textbf{2D}: \qquad \mathbf{D}_{xy} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Diese Faltungsmasken erhält man über die Diskretisierung des 2D-Laplace-Operators $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Die folgende Abb. 10.6 zeigt die Anwendung des Laplace-Filters auf ein Bild. Man sieht, dass jede Kante von einem schwarzen und weißen Rand umgeben sind (vgl. Abb. 10.5).

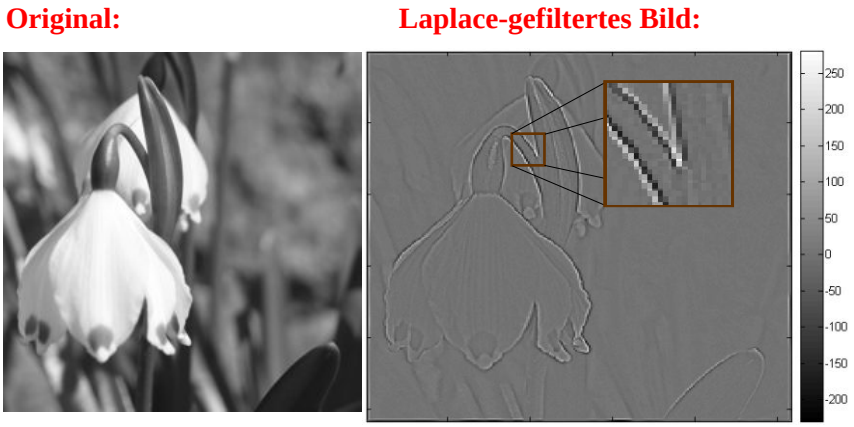


Abb. 10.6: Anwendung des Laplace-Filters auf ein Testbild

Transferfunktion: Der 2D-Filter hat folgende Transferfunktion:

$$\hat{l}(\vec{k}) = -4\sin^2(\pi k_x/2) - 4\sin^2(\pi k_y/2)$$

Die folgende Abb. 10.7 zeigt die Transferfunktion des Laplace-Filters. Es ist zu beachten, dass sie an den Rändern negativ wird. In der Mitte ist die 0. Diese Transferfunktion ist nur für kleine k isotrop.

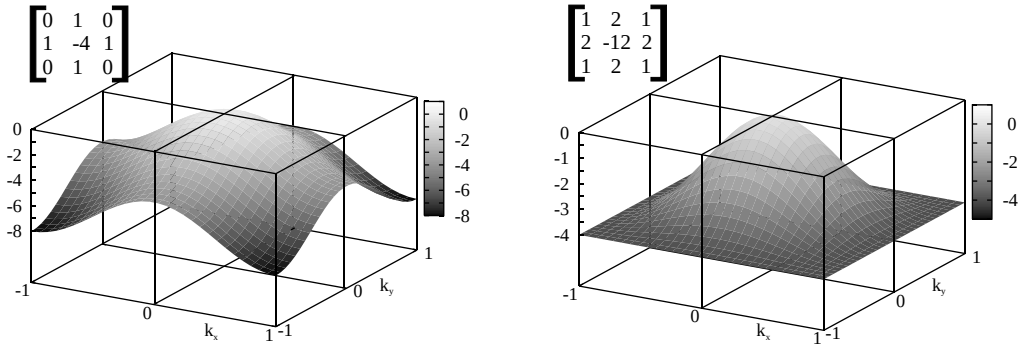


Abb. 10.7: Transferfunktion des einfachen Laplace-Filters

Man kann leicht erkennen, das niedrige Wellenzahlen herausgefiltert werden. Es handelt sich also um einen Hochpassfilter. Zusätzlich gilt $\hat{l}(k) \leq 0$. Damit erzeugt der Filter eine Phasenverschiebung von 180°.

Konstruktion über Binomialoperatoren: Ein Laplace-Operator lässt sich auch mit Hilfe von Binomialoperatoren konstruieren. Dies geschiet folgendermaßen:

$$4(\mathcal{B}^2 - \mathbb{1}) = \frac{1}{4}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} - \frac{1}{4}\begin{bmatrix}0 & 0 & 0\\ 0 & 16 & 0\\ 0 & 0 & 0\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1 & 2 & 1\\ 2 & -12 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Die Transferfunktion dieses Filters lautet:

$$\hat{l}(\vec{k}) = -4\cos^2(\pi k_x/2)\cdot\cos^2(\pi k_y/2) - 4$$

Diese Funktion ist wesentlich isotroper als die obige.

10.6 Optimierung der Kantenfilter

10.6.1 Allgemeine Optimierungsansätze

Least-Square-Fits: Man kann Least-Square-Fits benutzen, um die Filterkoeffizienten zu optimieren. Dazu nutzt man die allgemeine Form der Transferfunktion und bestimmt die optimalen Koeffizienten der Faltungsmaske so, dass diese der idealen Transferfunktion möglichst gut entsprechen. Das folgende Beispiel zeigt das für einen 1D-Gradientenfilter:

- **Zielfunktion:** $\hat{t}(k) = \mathrm{i}\pi k$
- **Transferfunktion:** $\hat{d}(k) = -2\mathrm{i}\cdot\sum\limits_{n'=1}^{R} h_{n'}\sin(n'\pi k)$

Man löst dann das Optimierungsproblem $\left(\hat{t}(k) - \hat{d}(k)\right)^2 \quad\rightarrow\quad \min.$

10.6.2 Regularisierte Kantendetektoren

Bei verrauschten Bildern liefern die bisherigen Detektoren nur schlechte Ergebnisse, weil siehe hohe Frequenzanteile und damit auch Rauschen verstärken. Dieses Problem lässt sich lösen, wenn man zur Kantendetektion noch eine Glättung hinzufügt. Bei einem solchen Ansatz ist es wichtig den so entstehenden Filter auf die Skalen der erwarteten Objekte im Bild anzupassen. Man muss den Filter so einstellen, dass er auf dem Wellenzahlbereich mit dem maximalen Signal-zu-Rausch-Verhältnis arbeitet. Das ist der Bereich, in dem das Signal am stärksten über das Rauschen hinausgeht. Man definiert deswegen:

Definition 10.1 (Regularisierter Filter) *Ein Ableitungsfilter, das eine Glättung enthält wird regularisiert genannt.*

Regularisierte Filter stellen eine robuste Lösung für das schlecht gestellte Problem der Ableitung diskreter, verrauschter Signale dar. Die Glättung muss so gestaltet werden, dass sie nur parallel zu Kanten glättet, nicht aber senkrecht, damit die Kanten nicht verschmiert werden.

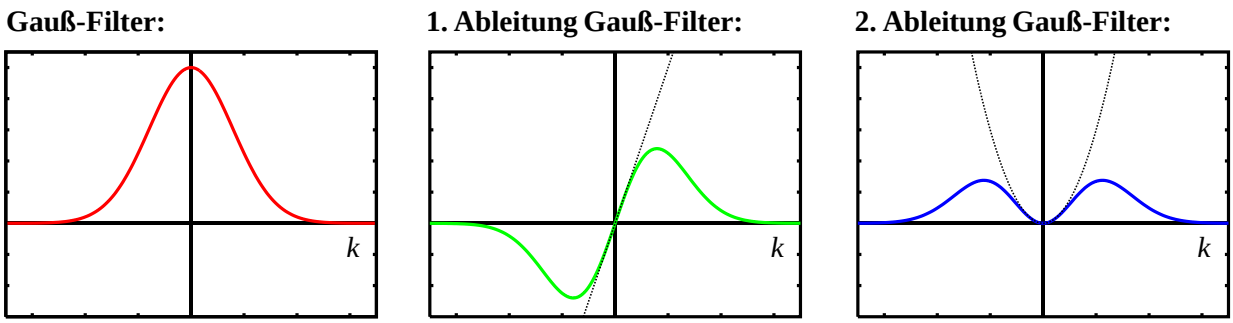


Abb. 10.8: Erste und Zweite Ableitung einer Gauß-Kurve. Man sieht die Ähnlichkeit zur idealen Transferfunktion von Gradienten- und Laplace-Operatoren.

2×2-Gradientenfilter: Aus den Gradienten- und Binomialfiltern lassen sich einfach regularisierte Filter konstruieren:

	Filtermaske	Transferfunktion
x-Richtung	$\mathcal{D}_x\mathcal{B}_y = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$	$\hat{d}_x\hat{b}_y(\vec{k}) = 2\mathrm{i}\sin(\pi k_x/2)\cdot\cos(\pi k_y/2)$
y-Richtung	$\mathcal{D}_y\mathcal{B}_x = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$	$\hat{d}_y\hat{b}_x(\vec{k}) = 2\mathrm{i}\sin(\pi k_y/2)\cdot\cos(\pi k_x/2)$

Diese Filter sind in Richtung der Ableitung wie einfache Gradientenfilter. Senkrecht dazu sind sie einfache Binomial-Glättungsfilter.

Sobel-Filter: Die Sobel-Gradientenfilter bestehen aus einer Gradientenbildung und einer stärkeren Glättung, als im letzten Absatz:

$$\mathcal{D}_{2x}\mathcal{B}_y^2 = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \mathcal{D}_{2y}\mathcal{B}_x^2 = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Canny-Filter, Canny-Kantendetektion: Diese Filter basieren auf den Ableitungen von Gaußfiltern. Dazu betrachtet man folgende Abb. 10.8. Die Transferfunktion eines Gaußfilters ist ebenfalls eine Gauß-Kurve. Die erste Ableitung dieser Gaußfunktion zeigt um $k = 0$ die Charakteristik eines Gradientenfilters ($\propto k$). Die zweite Ableitung ähnelt um $k = 0$ der Transferfunktion eines Laplace-Filters ($\propto k^2$). Man muss aus obigen Transferfunktionen also nur noch durch Fourier-Transformation die entsprechenden Filtermasken berechnen. In der realen Anwendung wird man wohl aber einfach zwei Filter wählen, die nacheinander angewendet werden. Ein solcher Filter ist optimal bei additivem Rauschen auf (maximal) linearen Kanten. Man kann ihn auch einfach so ableiten: Zunächst wird das Rauschen im Bild mit einem Gauß-Filter $\mathcal{G}$ geglättet. Nun wird ein Ableitungsfilter $\mathcal{D} = \frac{\partial}{\partial x}$ angewendet. Insgesamt hat man also für ein Bild g:

$$g' = \mathcal{D}\mathcal{G}g$$

Bei der Anwendung von LSI-Filter kann man beliebig klammern, sodass gilt:

$$g' = (\mathcal{D}\mathcal{G})g$$

Dies bedeutet, dass man auch zunächst die Ableitung des Glättungsfilters berechnen kann und dann den so entstehenden Filter auf das Bild anwenden.

Dieser Filter wird wohl als Canny-Filter bezeichnet. Mit Canny-Kantendetektion bezeichnet man einen spezielle Algorithmus zum Auffinden der Kanten im Bild. Man setzt dazu zunächst einen Schwellwert fest. Liegt der Gradientenbetrag an einem Pixel über diesem Schwellwert, so handelt es sich um eine Kante. Der Algorithmus veruscht nun den Kanten zu folgen. Dabei wird der Schwellwert heruntergesetzt, um auch schwächere Linien zu finden.

LoG-Filter: Die Laplace-of-Gaussian- oder LoG-Filter arbeiten nach dem selben Muster, wie Canny-Filter. Zunächst glättet man mit einem Gaußfilter (im diskreten einem Binomialfilter) und wendet dann den Laplace-Operator $\mathcal{L}$ an:

$$\mathcal{L}\mathcal{B}^p$$

DoG-Filter: Die Difference-of-Gaussian- oder DoG-Filter verfahren nach dem selben prinzip, wie die LoG-Filter. Es wird allerdings nicht die zweite Ableitung mit einem einfachen Laplace-Operator, sondern mit der Differenzdarstellung aus Binomialfiltern berechnet:

$$\underbrace{4(\mathcal{B}^q - \mathbb{1})}_{\text{Laplace}}\mathcal{B}^p$$

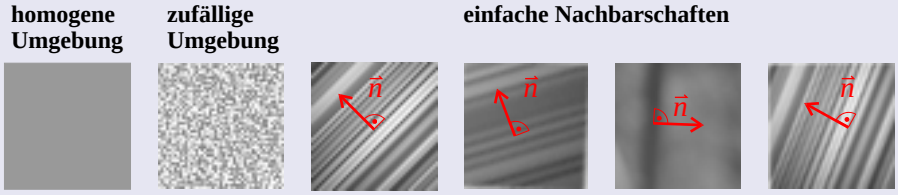
Ein solcher DoG-Filter hat eine geringere Anisotropie, als ein LoG-Filter. Man kann die Anisotropie noch weiter reduzieren, wenn man LoG und DoG-Filter kombiniert:

$$\frac{2}{3}\text{DoG} + \frac{1}{3}\text{LoG}$$

11 Einfache Nachbarschaften

11.1 Einführung

Definition 11.1 (einfache Nachbarschaft) *Eine einfache Nachbarschaft ist eine (kleine) lokale Umgebung in einem Bild, in der sich der Grauwert nur entlang einer Richtung ändert. Dies bedeutet, dass die Nachbarschaft nur aus parallelen Linien besteht. Die folgende Abbildung zeigt einige Beispiele:*



Dabei bezeichnet $\vec{n}$ den Normalenvektor auf die Grauwertstruktur, also die Richtung der Grauwertänderung. Man kann die Nachbarschaft so darstellen:

$$g(\vec{x}) \equiv g(\vec{x} \cdot \vec{n})$$

Dabei beschreibt $g(x)$ den Grauwert entlang der Normalenrichtung.

Die Richtigkeit der obigen Darstellung ergibt sich einfach durch die Berechnung des Gradienten:

$$\vec{\nabla} g(\vec{x} \cdot \vec{n}) = \vec{n} \cdot g'(\vec{x} \cdot \vec{n})$$

Dies bedeutet, dass die Änderung des Grauwertes nur in Richtung $\vec{n}$ bedeutet. QED.

Richtung und Orientierung: Es macht keinen Sinn die Richtung des Vektors $\vec{n}$ im vollen Winkelbereich $0..360°$ anzugeben, da ein um $180°$ gedrehtes Muster nur gespiegelt ist. Man gibt daher die Ausrichtung der Nachbarschaft als sog. **Orientierung** an, die nur zwischen $0°$ und $180°$ liegt. Eine Ausrichtung im vollen Winkelbereich $0..360°$ wird als **Richtung** bezeichnet.

Fourier-Darstellung: Im Fourierraum steht jeder Pixel $\vec{k}$ für eine periodische Struktur mit bestimmter Ausrichtung entlang $\vec{k} = (k_x, \quad k_y)^t$ (siehe Abb. 3.7). Damit muss der Wellenvektor $\vec{k}$ jeder Fourier-Komponente, die in der Nachbarschaft vorkommt senkrecht auf dem Vektor $\vec{n}$ der Nachbarschaft stehen. Die Fourier-Transformierte einer einfachen Nachbarschaft ist also eine Linie entlang $\vec{n} \parallel \vec{k}$:

$$g(\vec{x} \cdot \vec{n}) \quad \circ\!\!-\!\!\bullet \quad \hat{g}(\vec{k}) \cdot \underbrace{\delta\big[\vec{k} - (\vec{k} \cdot \vec{n})\vec{n}\big]}_{\text{schränkt auf Linie ein}}$$

Bisher wurde davon ausgegangen, dass sich die einfache Nachbarschaft unendlich ausdehnt. Es soll sich aber um einen kleinen Bereich handeln (eben eine Nachbarschaft), sodass sie durch eine Fensterfunktion $w(\vec{x})$ beschränkt werden muss (z.B. Gauß-Fenster). Das Bild $g(\vec{x} \cdot \vec{n})$ wird also mit der Fensterfunktion multipliziert:

$$g(\vec{x}) = w(\vec{x}) \cdot g(\vec{x} \cdot \vec{n})$$

Im Fourier-Raum bedeutet dies eine Faltung mit $\hat{w}(\vec{k})$.

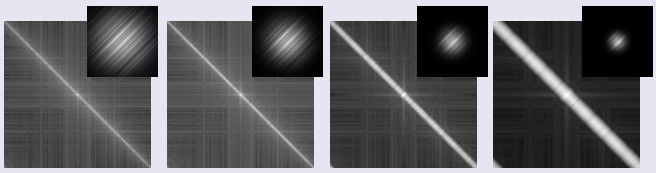
Satz 11.1 (Fourier-Transformierte einer einfachen Nachbarschaft) *Eine einfache Nachbarschaft, deren Grauwert sich nur entlang $\vec{n}$ ändert hat folgende Darstellung in Real- und Fourierraum:*

$$w(\vec{x}) \cdot g(\vec{x} \cdot \vec{n}) \quad \circ\!\!-\!\!\bullet \quad \hat{w}(\vec{k}) \circledast \hat{g}(\vec{k}) \cdot \delta\big[\vec{k} - (\vec{k} \cdot \vec{n})\vec{n}\big]$$

Die Fourier-Darstellung einer unbeschränkten Nachbarschaft ist also eine Linie. Die Fensterfunktion $w(\vec{x})$ definiert die Größe der Nachbarschaft. Im Fourierraum wird die Linie mit $\hat{w}(\vec{k})$ gefaltet, was eine Verbreiterung der Linie zur Folge hat. In der folgenden Abbildung gibt σ die Breite des Nachbarschaft an.

Größe der Nachbarschaft: σ = 70 σ = 50 σ = 30 σ = 20

Fourier-Transformierte:



11.2 Vektordarstellung

Um eine lokale Nachbarschaft vollständig zu beschreiben muss man neben ihrer Orientierung auch noch ein Bestimmtheitsmaß angeben, das angibt, wie stark die Nachbarschaft tatsächlich orientiert ist. Ist das Bestimmtheitsmaß 1, so ist die Nachbarschaft perfekt orientiert. Ist es 0, so ist die Umgebung ohne erkennbare Orientierung (z.B. verrauscht oder homogen). Man fasst diese zwei Größen zu einer vektoriellen Größe zusammen:

Definition 11.2 (Vektordarstellung lokaler Nachbarschaften) *Zur Beschreibung einer lokalen Nachbarschaft werden zwei Größen herangezogen: die Orientierung $\vec{n}$ der Nachbarschaft und ein Bestimmtheitsmaß für die Stärke der Orientierung. Der darstellende Vektor zeigt in die Richtung von $\vec{n}$. Seine Länge ist das Bestimmtheitsmaß. Je länger der Vektor ist, desto besser ist die Orientierung bestimmt.*

Diese Definition ermöglicht es über mehrere Nachbarschaften zu mitteln, indem man über mehrere darstellenden Vektoren addiert. Je bestimmter (länger) ein solcher Vektor ist, desto mehr trägt er zur Summe bei. Man kann für einen solchen Orientierungsvektor eine einfache Farbdarstellung finden. Sein Betrag wird als Helligkeit und seine Richtung als Farbwert kodiert. Die folgende Abbildung zeigt einen Halbkreis der in Farben kodiert ist:

Man kann dann aus einem Grauwertbild ein Farbbild berechnen. Es zeigt nur dort sichtbare Farben, wo eine gut bestimmte orientierte Nachbarschaften vorliegt. Die dortige Farbe zeigt die Ausrichtung der Nachbarschaft an.

Diese Darstellung hat allerdings einen Nachteil: Man kann mit ihr nicht zwischen homogener und zufälliger Nachbarschaft unterscheiden. Beide führen zu einem sehr kurzen Vektor. Man wählt deswegen andere Darstellungen, wie die folgenden Tensordarstellungen.

11.3 Tensordarstellung erster Ordnung

11.3.1 Strukturtensor

Es sollen nun etwas bessere Darstellungen untersucht werden, als die Vektordarstellung aus Abschnitt 11.2. Man betrachtet hierfür folgende Optimierungsaufgabe:

Finde ein $\vec{n}$, das möglichst gut zu einer gegebenen Nachbarschaft passt.

Um diese Optimierungsaufgabe mathematisch zu formulieren muss man ein Maß finden, das die Abweichung der zu optimierenden Größe $\vec{n}$ von der tatsächlichen Ausrichtung (bestimmt über den Gradienten $\vec{\nabla} g$ des Grauwertes) in der Nachbarschaft angibt. Man wählt hier:

$$\epsilon = \left[(\vec{\nabla} g) \cdot \vec{n}\right]^2 = \left|\vec{\nabla} g\right|^2 \cdot \cos^2\left[\angle(\vec{\nabla} g, \vec{n})\right]$$

Dabei ist $\angle(\vec{\nabla} g, \vec{n})$ der Winkel zwischen $\vec{n}$ und der tatsächlichen Orientierung. ϵ wird maximal, wenn $\vec{n}$ und $\vec{\nabla} g$ parallel liegen. Man muss also zur Bestimmung von $\vec{n}$ das Fehlermaß ϵ maximieren. Dieses Maß berücksichtigt noch nicht die endliche Ausdehnung der Nachbarschaft (beschrieben durch das Fenster $w(\vec{x})$). Um dieses ebenfalls zu berücksichtigen faltet man das Fehlermaß mit $w(\vec{x})$ und maximiert somit den folgenden Ausdruck:

$$\int_{-\infty}^{\infty} \left[\left(\vec{\nabla} g(x')\right) \cdot \vec{n}\right]^2 \cdot w(\vec{x} - \vec{x}')\, \mathrm{d}^W x' \quad \rightarrow \quad \max$$

Man kann dies mit dem sog. Strukturtensor $\mathbf{J}$ (einer Matrix) auch so darstellen (nach einigen Umformungen):

$$\vec{n} \cdot \left(\mathbf{J}\vec{n}\right) \quad \rightarrow \quad \max$$

Dabei wurde ausgenutzt, dass man $\left[\left(\vec{\nabla} g(x')\right) \cdot \vec{n}\right]^2$ auch so darstellen kann:

$$\left[\left(\vec{\nabla} g(x')\right) \cdot \vec{n}\right]^2 = \left[\left(\vec{\nabla} g(x')\right) \cdot \vec{n}\right] \cdot \left[\left(\vec{\nabla} g(x')\right) \cdot \vec{n}\right] = \vec{n}^t (\vec{\nabla} g \vec{\nabla} g^t) \vec{n}$$

Der Strukturtensor ist dann so definiert:

$$\boxed{J_{pq}(\vec{x}) = \int_{-\infty}^{\infty} w(\vec{x} - \vec{x}') \cdot \left(\frac{\partial g(\vec{x}')}{\partial x'_p} \cdot \frac{\partial g(\vec{x}')}{\partial x'_q}\right) \mathrm{d}^W x'}$$

Man spricht hier von einer Darstellung erster Ordnung, weil zum einen nur Ableitungen erster Ordnung vorkommen und man zum Anderen nur einfache Nachbarschaften betrachtet. Der Tensor $\mathbf{J}$ ist symmetrisch. Somit kann man das Koordinatensystem so drehen, dass er Diagonalgestalt hat. Die zugehörigen Basisvektoren $\vec{e}_i$ sind gerade die Eigenvektoren zu den Eigenwerten λ_i von $\mathbf{J}$ (Eigenwertgleichung: $\mathbf{J}\vec{e}_i = \lambda_i \vec{e}_i$). Ist der Tensor in Diagonalgestalt, so liegt eine Koordinatenachse parallel zu $\vec{n}$. Damit genügt es also den Drehwinkel θ auszurechnen, um $\vec{n}$ zu erhalten. Die Rotation einer Matrix erfolgt mit einer unitären Transformationsmatrix $\mathbf{U} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$ nach der Vorschrift $\mathbf{J}' = \mathbf{U}^t \mathbf{J} \mathbf{U}$:

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = \mathbf{U}^t \mathbf{J} \mathbf{U} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Nach längerer Rechnung ergibt sich daraus:

$$\boxed{\tan 2\theta = \frac{2 \cdot J_{12}}{J_{22} - J_{11}}}$$

Daraus wiederum ergibt sich der Orientierungsvektor $\vec{\sigma} = \begin{pmatrix} J_{22} - J_{11} \\ 2 \cdot J_{12} \end{pmatrix}$. Seine Länge kann als Bestimmtheitsmaß aufgefasst werden.

Aus der Analyse der Eigenwerte λ_i des Strukturtensors lassen sich ebenfalls Aussagen über die Nachbarschaft gewinnen:

	$\mathrm{Rang}(\mathbf{J})$	**Bedeutung**
$\lambda_1 = \lambda_2 = 0$	0	konstante Umgebung
$\lambda_1 > 0, \ \lambda_2 = 0$	1	einfache Nachbarschaft
$\lambda_1 > 0, \ \lambda_2 > 0$	2	Grauwerte ändern sich in alle Richtungen. Bei $\lambda_1 = \lambda_2$ liegt eine isotrope Struktur vor

Die Größe $J_{11} + J_{22} = \mathrm{Spur}\,\mathbf{J}$ ist ein Maß für die Länge des Gradienten. Man kann deshalb ein sog. **Kohärenzmaß** c_c definieren. Für $c_c = 0$ liegen isotrop verteilte Grauwerte vor, für $c_c = 1$ hat man eine ideale Orientierung. Es gilt:

$$c_c = \frac{|\vec{\sigma}|}{|\vec{\nabla} g|} = \frac{\sqrt{(J_{22} - J_{11})^2 + 4J_{12}^2}}{J_{11} + J_{22}} = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}$$

Farbdarstellung: Auch den Strukturtensor kann man wieder als Farbbild darstellen:

- Orientierung $\theta \rightarrow$ Farbwert
- Länge des Gradienten $\vec{\nabla} g \rightarrow$ Helligkeit
- Kohärenzmaß $c_c \rightarrow$ Sättigung

Die Länge des Gradienten gibt die Stärke der Grauwertänderung in der Nachbarschaft an. Das Kohärenzmaß gibt an, ob es sich um eine isotrope oder eine geordnete Nachbarschaft handelt.

Implementierung: Die Komponenten J_{pq} des Strukturtensors lassen sich leicht berechnen. Das Integral beschreibt nur die Faltung mit einer Fensterfunktion $w(\vec{x})$. Handelt es sich hierbei um ein gauß'sches Fenster, so kann man dieses Integral durch einen Binomialfilter $\mathcal{B}$ entsprechender Breite implementieren. Zur Berechnung der zwei Ableitungen in p- und q-Richtung verwendet man Ableitungsfilter $\mathcal{D}_p, \mathcal{D}_q$. Die Operation $\cdot$ beschreibt eine pixelweise Multiplikation, sodass man erhält:

$$\mathcal{J}_{pq} = \mathcal{B}(\mathcal{D}_p \cdot \mathcal{D}_q)$$

Bei der Berechnung der Eigenwerte kann man auf gut bekannte Methoden der numerischen Mathematik zurückgreifen. Solange nur 2D-Bilder vorliegen kann man die Gleichungen sogar per Hand lösen und die entsprechenden Formeln im Programm verwenden. Die Glättung benötigt in obiger Formel die meisten Operationen, sodass es sinnvoll erscheint J_{pq} nicht auf dem vollen Gitter des Bildes g zu berechnen, sondern vorher auf ein kleineres Gitter zu reduzieren. Dabei geht keine Information verloren, weil ja noch eine Glättung stattfindet. Außerdem sieht man hier das allgemeine Prinzip, dass bei der Berechnung höherer Informationen aus einem Bild immer Auflösung verlorengeht. Für die Ableitungsoperatoren sollte man optimierte Operatoren verwenden, um den Winkelfehler so klein wie möglich zu halten.

Wichtig ist noch, bei der Klassifizierung der Eigenwerte nicht auf Gleichheit zu 0 zu prüfen, sondern ob die Zahl unterhalb einer kleinen Schwelle $s_0 \ll 1$ liegt, da numerische Verfahren wegen der Rundungsfehler fast nie exakt 0 ergeben.

Das folgende Listing zeigt eine Implementierung des Strukturtensors in Matlab.

Listing 11.1: Berechnung des Strukturtensors in Matlab

```
# Bild in A laden
A = double(imread('textur2.png'));

# Filtermasken definieren, Sobel-Filter in x- und y-Richtung,
# sowie Binomial-Maske
sobelx=[1 0 -1,
        2 0 -2,
        1 0 -1]
sobely=[ 1  2  1,
         0  0  0,
        -1 -2 -1]
binomial = 1/16*[ 1 2 1,
                  2 4 2,
                  1 2 1]

# Ableitungen berechnen
dx=conv2(A,sobelx);
dy=conv2(A,sobely);

# Strukturtensor-Komponenten berechnen
;
J11=conv2(dx.*dx,binomial);
J12=conv2(dx.*dy,binomial);
J22=conv2(dy.*dy,binomial);

# Winkel theta und Kohärenzmaß cc berechnen
theta=atan2(2*J12,(J22-J11))+pi;
cc=sqrt((J22-J11).*(J22-J11)+4*J12.*J12)./(J11+J22);
```

In der folgenden Abb. 11.1 sind dann die Ergebnisse dieses Programms gezeigt. Man sieht deutlich, wie die Farbe den Winkel θ codiert. Die unterschiedlich ausgerichteten Bereiche des Bildes werden gut getrennt. Im verrauschten Testbild-Teil ist auch der Strukturtensor verrauscht und das Kohärenzmaß ist eher klein (dunkel). Dies bedeutet, dass hier keine gute Ausrichtung vorliegt. Die schwarze Umgebung der Testfelder erhält einen einheitlichen Winkel. Das Kohärenzmaß in dieser homogenen Nachbarschaft ist 0. Das Kohärenzmaß ist in den einzelnen Felder überall fast 1, was eine perfekte Orientierung anzeigt. An den grenzen zwischen zwei verschiedenen Orientierungen wird das Kohärenzmaß kleiner (siehe zweites Feld mit Unterfeld). Der Kreis zeigt, in welche Farben die Orientierung codiert wird und dass Winkel, die sich um 180° unterscheiden auf den selben Wert abgebildet werden.

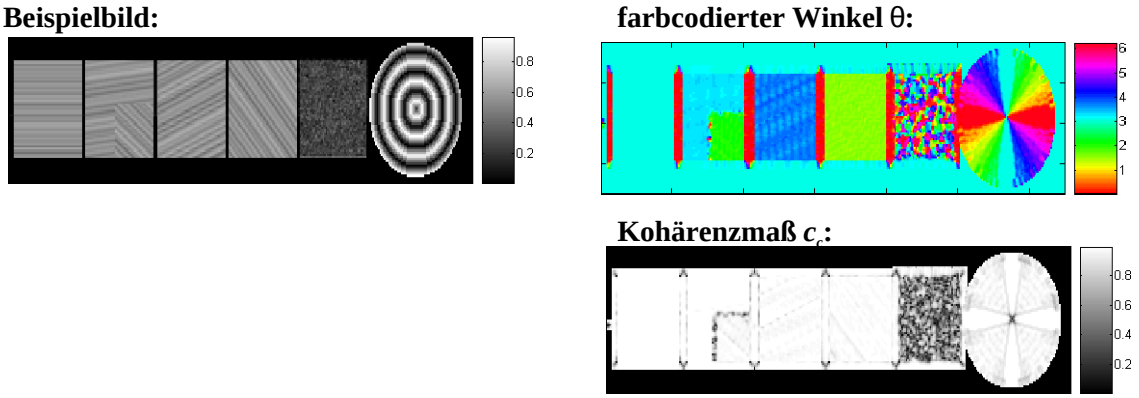


Abb. 11.1: Berechnung des Strukturtensors eines Beispielbildes

11.3.2 Andere Tensordarstellungen

Neben dem Strukturtensor kann man auch andere Tensordarstellungen für lokale Nachbarschaften finden. Wie im vorherigen Abschnitt gewinnt man diese durch eine Optimierungsaufgabe, allerdings mit einer anderen Optimierungsfunktion. Man geht wieder davon aus, dass die Fourier-Transformierte der Nachbarschaft eine Linie im Fourier-Raum bildet. Man berechnet dann die Regressionsgerade durch die

Punkte des Fourier-Raums, wobei ein Punkt $\vec{k}$ umso stärker gewichtet wird, je größer sein Betrag $|\hat{g}(\vec{k})|^2$ ist. Dies ist sinnvoll, da ja nur Punkte auf der Gerade von 0 verschieden sein sollten. Also tragen auc nur Punkte nahe der tatsächlichen gerade zur Regression bei. Dieses Vorgehen entspricht demjenigen, dass man zunächst alle Punkte findet, die auf der Geraden im Fourier-Raum liegen und diese zur Berechnung der Regressionsgeraden heranzieht. Aus den Parametern der Regressionsgerade kann man dann $\vec{n}$ bestimmen. Die folgende Abb. 11.2 veranschaulicht das.

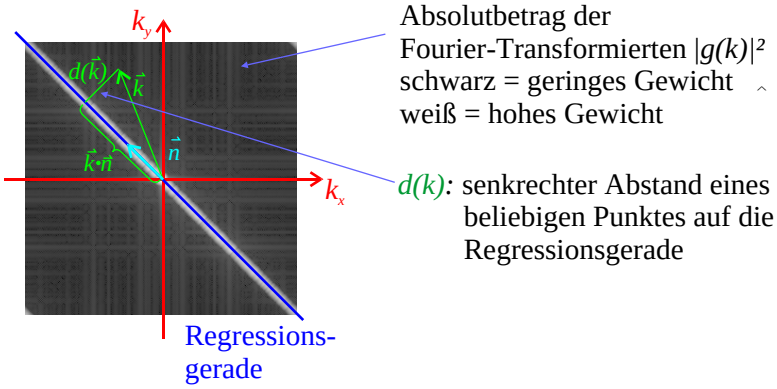


Abb. 11.2: Regressionsgerade an die Fourier-Transformierte einer einfachen Nachbarschaft

Der Abstand $|\vec{d}(\vec{k})|^2$ eines beliebigen Punktes $\vec{k}$ im Fourier-Raum zur Regressionsgerade wird aus folgender Gleichung berechnet (Satz des Pythagoras!):

$$d^2(\vec{k}) + (\vec{k}\cdot\vec{n})^2 = |\vec{k}|^2$$

Der Vektor $\vec{n}$ zeigt entlang der Regressionsgerade. Der Ausdruck $\vec{k}\cdot\vec{n}$ stellt den Anteil von $\vec{k}$ dar, der parallel zu $\vec{n}$ liegt. Es wird dann die gewichtete Summe über alle diese Abstände im Fourier-Raum minimiert:

$$\int_{-\infty}^{\infty} |\hat{g}(\vec{k})|^2 \cdot d^2(\vec{k},\vec{n})\ \mathrm{d}^W k \quad \rightarrow \quad \min$$

Bei der Minimierung dieses Integrals tritt der Trägheitstensor **I** auf. Aus seiner Eigenwertanalyse kann man wieder auf die Art der Nachbarschaft schließen. Für den Trägheitstensor **I** gilt:

$$|d|^2 = |\vec{k}|^2 - |\vec{k}\cdot\vec{n}|^2 = \vec{n}^t\left[\mathbb{1}(\vec{k}\cdot\vec{k}) - \vec{k}\vec{k}^t\right]\vec{n} = \vec{n}^t\mathbf{I}\vec{n} \quad \rightarrow \quad \min$$

Die Minimierung dieses Ausdruckes ist wieder dem obigen Integral äquivalent, weil die Integration vollständig in den Tensor **I** hineingezogen werden kann. Man erhält:

$$I_{pp} = \sum_{q\neq p}\int_{-\infty}^{\infty} k_q^2 |\hat{g}(\vec{k})|^2\ \mathrm{d}^W k$$

$$I_{pq} = -\int_{-\infty}^{\infty} k_p k_q |\hat{g}(\vec{k})|^2\ \mathrm{d}^W k, \quad \text{für } p \neq q$$

11.4 Lokale Wellenzahl und Phase

11.4.1 Hilbertfilter und Hilberttransformation

Zur Einführung betrachte man ein einfaches Cosinus-Signal:

$$g(x) = g_0 \cdot \cos(\underbrace{k \cdot x}_{=\phi(x)})$$

Dabei bezeichnet g_0 die Amplitude, k die Wellenzahl und $\phi(x)$ die lokale Phase des Bildes. Die Wellenzahl k lässt sich leicht aus der Phase berechnen:

$$k = \frac{\partial \phi(x)}{\partial x}$$

Genauso wie die Phase $\phi(x)$ vom Ort abhängt kann nun auch die Wellenzahl k vom Ort abhängen. Man spricht dann von einer lokalen Wellenzahl $k(x)$, die komplexere Bilder ermöglicht, weil jetzt die Phase nicht mehr rein linear steigen muss. Die folgende Abb. 11.3 zeigt ein Beispiel.

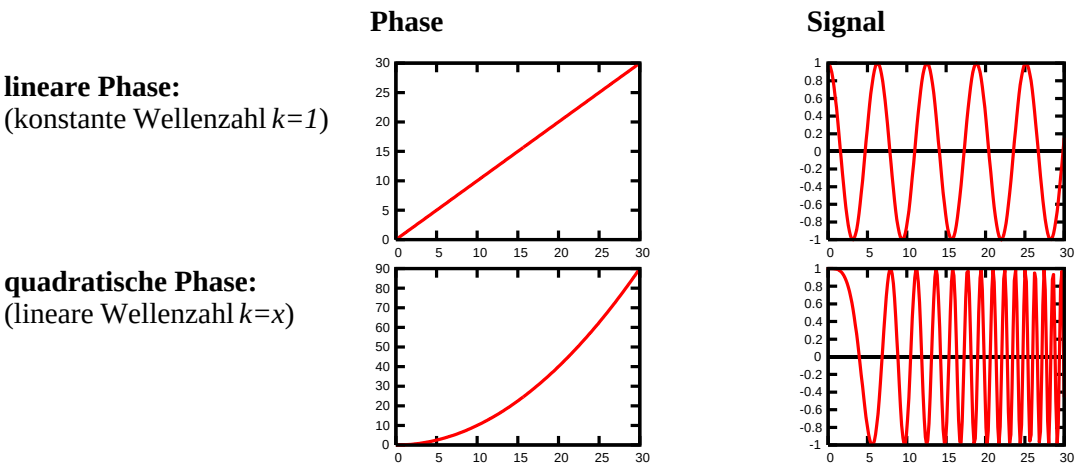


Abb. 11.3: Beispiel für lokalen Wellenzahlen und Phasen

Es stellt sich nun die Frage, wie man die lokale Phase (und damit auch die lokale Wellenzahl) eines Bildes berechnen kann, wenn man nur das Bild g gegeben hat. Dazu geht man nochmals von $g(x) = g_0 \cdot \cos\phi(x)$ aus. Wenn man dieses Signal mit einem Filter $\mathcal{H}$ um $90°$ oder $\pi/2$ phasenverschiebt , so erhält man:

$$g'(x) = \mathcal{H}g(x) = -g_0 \cdot \sin\phi(x)$$

Diese Operation entspricht (nur für Sinus- und Cosinus-Funktionen) der Berechnung der 1. Ableitung. Nun ist der Tangens ganz allgemein definiert als

$$\tan x = \frac{\sin x}{\cos x}$$

Damit kann man aber mit obigen speziellen Bildern g und g' auch schreiben:

$$\tan(kx) = \frac{\sin(kx)}{\cos(kx)} = \frac{-g'(x)}{g(x)}$$

Dieser Zusammenhang lässt sich für beliebige Bilder $g(x)$ verallgemeinern:

Satz 11.2 (Lokale Phase und Wellenzahl) *Sei g ein beliebiges Bild und $\mathcal{H}$ ein Operator, der das Bild um $90°$ phasenverschiebt. Er muss also alle Fourier-Komponenten des Bildes um $90°$ verschieben. Man kann dann die lokale Phase $\phi(x)$ des Bildes berechnen durch:*

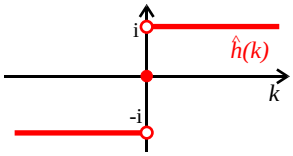
$$\phi(x) = \arctan\left(\frac{-\mathcal{H}g(x)}{g(x)}\right)$$

Aus der lokalen Phase $\phi(x)$ kann man die lokale Wellenzahl durch Differenzieren (Gradientenoperator $\mathcal{D}_x$) berechnen:

$$k(x) = \frac{\partial\phi(x)}{\partial x} = \mathcal{D}_x\phi(x)$$

Hilbert-Filter: Ein Filter $\mathcal{H}$, das eine $90°$ Phasenverschiebung berechnet heißt Hilbert-Filter. Seine ideale Transferfunktion ist gegeben als:

$$\hat{h}(k) = \begin{cases} \mathrm{i} & k > 0 \\ 0 & k = 0 \\ -\mathrm{i} & k < 0 \end{cases}$$



Die Punktantwort $h(x)$ dieser Stufenfunktion ist wohl bekannt:

$$h(x) = -\frac{1}{\pi x}$$

Man kann dann die Anwendung (Faltung) des idealen Hilbert-Filters so schreiben:

$$\mathcal{H}g(x) = h \circledast g = \int_{-\infty}^{\infty} h(x - x') \cdot g(x')\,\mathrm{d}x' = -\frac{1}{\pi}\int_{-\infty}^{\infty} \frac{1}{x - x'} \cdot g(x')\,\mathrm{d}x'$$

Die durch dieses Integral definierte Integral-Transformation wird auch als Hilbert-Transformation bezeichnet.

Da die Punktantwort des ideale Hilbertfilter damit unendlich ausgedehnt ist, kann man keine kleine, diskrete Faltungsmaske angeben, die ein exaktes Hilbert-Filter darstellt. Deswegen schränkt man die Klasse der Signale ein, die man mit einem Hilbert-Filter bearbeiten kann. Wenn man nur band-begrenzte Signale (wie z.B. in einer Laplace-Pyramide) vorliegen hat, so kann man mit einem Least-Square-Fit ein Hilbert-Filter optimieren, das in dem Frequenzband $[k - \Delta k..k + \Delta k]$ des gegebenen Bildes nahezu ideal arbeitet. Dies vermeidet auch die Diskontinuität bei $k = 0$. Bei der Optimierung ist darauf zu achten, das die Phasenverschiebung nicht von der Wellenzahl abhängen darf.

Hilbert-Transformation in höherdimensionalen Signalen: Die Hilbert-Transformation funktioniert in der angegebenen Form nur auf 1D-Signalen. Für höherdimensionale Signale ist sie nicht definiert.

11.4.2 Analytisches Signal

Definition 11.3 (Analytisches Signal) *Fasst man ein Bild g mit seiner Hilbert-Transformation $g_h = \mathcal{H}g$ auf folgende Weise zusammen, so nennt man das so definierte neue Signal g_a als analytisches Signal:*

$$g_a = \mathcal{A}g = g - \mathrm{i} \cdot g_h = g - \mathrm{i}\mathcal{H}g$$

Der Operator $\mathcal{A}$ berechnet g_a aus einem Signal g.

Das analytische Signal g_a ist eine andere Darstellung des reellen Signals $g \in \mathbb{R}$. Sie enthält keine zusätzlichen Informationen. Mit dieser Darstellung ist es aber sehr viel einfach die lokale Phase und Amplitude des Signals g zu berechnen:

- lokale Phase: $\phi(x) = \arg(g_a)$
- lokale Amplitude: $g_0(x) = |g_a|$

Dabei ist das Argument $\arg(\cdot)$ einer komplexen Zahl so definiert:

$$z = r \cdot \mathrm{e}^{\mathrm{i}\varphi} \quad \Rightarrow \quad \varphi = \arg(z)$$

Außerdem kann man das Signal selbst und seine Hilbert-Transformation aus g_a berechnen:

$$g = \frac{g_a + g_a^*}{2} \qquad g_h = \mathcal{H}g = \frac{g_a - g_a^*}{2}$$

Erzeugung mit Filter $\mathcal{A}$: Das analytische Signal g_a wird erzeugt, indem man ein Filter $\mathcal{A}$ mit der Punktantwort

$$a(x)1 + \mathrm{i}h(x) = 1 + \frac{\mathrm{i}}{\pi x}$$

verwendet. Dabei ist $h(x)$ die Transferfunktion des Hilbertfilters $\mathcal{H}$ (siehe Abschnitt 11.4.1). Dieses Filter hat folgende Transferfunktion:

$$\hat{a}(k) = \begin{cases} 2 & k > 0 \\ 1 & k = 0 \\ 0 & k < 0 \end{cases}$$

11.4.3 Quadraturfilter

Neben den Hilbert-Filtern gibt es noch andere Möglichkeiten $90°$ Phasenverschobene Signale zu erzeugen. Man nutzt dazu ein Filter $\mathcal{Q}$ mit der folgenden Transferfunktion:

$$\hat{q}(\vec{k}) = \begin{cases} 2 \cdot h(\vec{k}) & \vec{k} \cdot \vec{n} > 0 \\ 0 & \text{sonst} \end{cases}$$

Dabei gibt $\vec{n}$ die Richtung an, in der das Filter angewendet wird und $h(\vec{k})$ ist eine beliebige reelle Funktion, die eine Wellenzahlwichtung darstellt und anwendungsspezifisch bestimmt wird. Das Filter $\mathcal{A}$ zur Erzeugung des analytischen Signals kann als Spezialfall dieser Quadraturfilter angesehen werden. Man kann mit dem Filter $\mathcal{Q}$ zwei neue Filter konstruieren:

$$\mathcal{Q}_+ : \quad \hat{q}_+(\vec{k}) = \frac{\hat{q}(\vec{k}) + \hat{q}(-\vec{k})}{2} \qquad \mathcal{Q}_- : \quad \hat{q}_-(\vec{k}) = \frac{\hat{q}(\vec{k}) - \hat{q}(-\vec{k})}{2}$$

Das besondere an diesen Filtern ist, dass die Signale $\mathcal{Q}_+ g$ und $\mathcal{Q}_- g$ um $90°$ zueinander Phasenverschoben sind. Zusätzlich sind beide noch um einen Phasenwinkel α verschoben, der aber bei beiden Signalen gleich ist. Damit kann man diese Signale verwenden, um die lokale Phase des Signals g zu berechnen:

$$\phi(\vec{x}) = \arctan\left(\frac{-p(\vec{x})}{q(\vec{x})}\right)$$

Dabei sind p und q die Signale, die durch Anwendung von $\mathcal{Q}_\pm$ auf g entstehen. Der Vorteil dieser Filter ist, dass sie auch für mehrdimensionale Signale definiert sind (im Gegensatz zur Hilber-Trafo).

Lokale Wellenzahl: Wie bereits erwähnt bildet erhält man die lokale Wellenzahl als erste Ableitung der lokalen Phase $\phi(x)$. Man kann auch einen exlpiziten Ausdruck angeben, mit dem man die lokale Wellenzahl direkt aus p und q berechnen kann:

$$\vec{k} = \vec{\nabla}\phi = \vec{\nabla}\arctan\left(\frac{-p(\vec{x})}{q(\vec{x})}\right) = \frac{q\vec{\nabla}p - p\vec{\nabla}q}{p^2+q^2}$$

Gaborfilter: Das Gaborfilter ist eines der bekanntesten Quadraturfilter. Es nutzt

$$h(\vec{k}) = \exp\left(-\frac{\left|\vec{k}-\vec{k}_0\right|^2 \cdot \sigma_x^2}{2}\right).$$

daraus ergibt sich dann folgende Punktantwort:

$$g(\vec{x}) = \mathrm{e}^{\mathrm{i}\vec{k}_0\vec{x}} \exp\left(-\frac{\left|\vec{x}\right|^2}{2\sigma_x^2}\right)$$

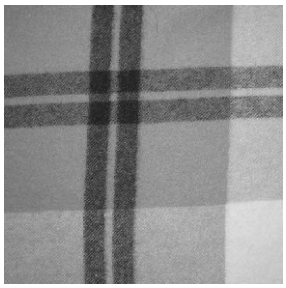
Dabei ist $\vec{k}_0$ $(= \vec{n})$ die Schwerpunktwellenlänge des Filters.

12 Textur

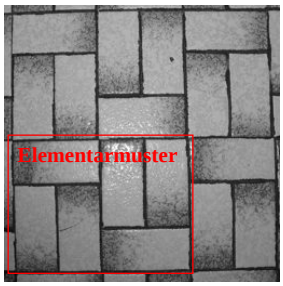
12.1 Einleitung

Definition 12.1 (Textur) *Die Textur bezeichnet das Füllmuster einer Fläche in einem Bild. Eine solche texturierte Fläche ist mit einem sich (periodisch oder quasiperiodisch) wiederholenden kleinen Muster gefüllt. Zur Beschreibung einer texturierten Fläche genügt also die Beschreibung des Elementarmusters und der Wiederholungseigenschaften. Die textur kann evtl. noch von der Zoomstufe abhängen.*
*Als **charakteristische Größe** einer Textur bezeichnet man die Größe des Elementarmusters.*

verschieden texturierte Bereiche:



Fußboden mit großem Elementarmuster:



Abhängigkeit der Textur von der Zoomstufe:

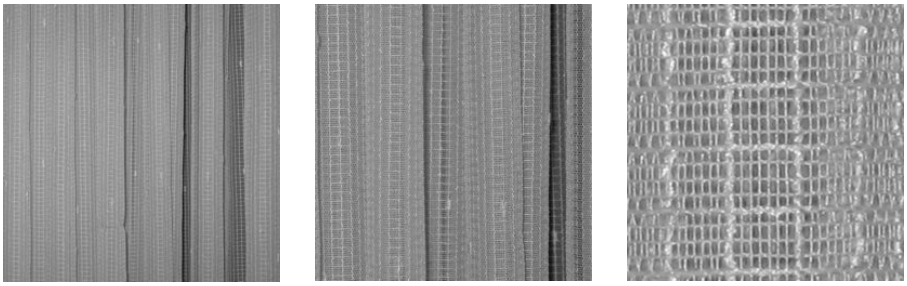


Abb. 12.1: verschiedene Texturen

Die Abb. 12.1 zeigt Beispiele für Texturen. In den unteren Bildern steht erst im rechten Bild die Fadenstruktur der Gardine als Textur im Vordergrund. Vorher spielen auch die Falten eine Rolle und die Fadenstruktur ist nur wenig sichtbar.

Die Parameter, die zur Beschreibung einer Textur dienen sollten unabhängig von Ausrichtung und Vergrößerungsstufe sein, da sonst die Klassifikation der textur von der Ausrichtung und vom Abstand der Kamera abhängen. Sind Texturen hierarchisch (wie etwa die Gardine im obigen Bild), so ist der zweite Aspekt nicht erfüllbar. Die Erfüllung des ersten Aspektes wird nicht immer gewünscht. So möchte man z.B. unterschiedlich ausgerichtete Streifenmuster manchmal durchaus unterscheiden.

In diesem Abschnitt sollen nun einige Verfahren besprochen werden, mit denen Texturen beschrieben werden sollen. Sie verwenden zur Beschreibung die statistischen Parameter Mittelwert und Varianz, sowie die Orientierung (siehe Abschnitt 11 über einfache Nachbarschaften) und Größe der Muster.

12.2 Statistik erster Ordnung

Die Statistik erster Ordnung für Texturen basiert auf den lokalen Grauwerthistogrammen. Sie ist damit unabhängig von Orientierung und Größe, da bei der Berechnung des Histogramms jegliche Information über die Position der Pixel verlorengeht. Sie ist damit rotations- und skaleninvariant. Dafür können bestimmte Muster nicht unterschieden werden, wenn sie das gleiche Histogramm aufweisen. So haben etwa die Texturen in Abb. 12.2 alle das gleiche Histogramm (ausgewogene bimodale Verteilung), obwohl sie sehr unterschiedlich aussehen.

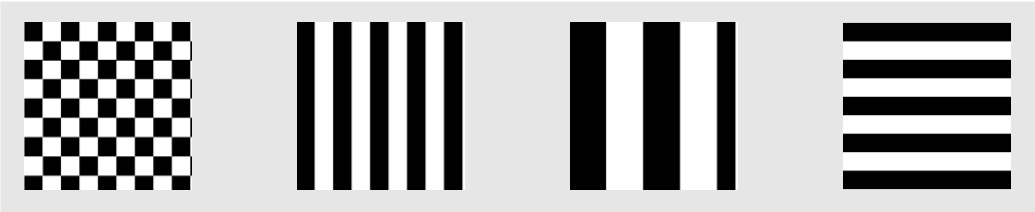


Abb. 12.2: Verschiedene Texturen mit dem selben Grauwerthistogramm

Beschreibung der Textur: Zur Beschreibung einer Textur mit Statistik erster Ordnung, betrachtet man einen Ausschnitt $\mathbb{M}$ des Bildes (maskiertes Bild). Man kann berechnen:

- **Mittelwert in der Umgebung $\mathbb{M}$ um $\vec{x} = (m, n)^t$:**

$$\overline{g}_{mn} = \frac{1}{P} \sum_{(m',n') \in \mathbb{M}} g_{m-m',n-n'}$$

- **Varianz in der Umgebung $\mathbb{M}$ um $\vec{x} = (m, n)^t$:**

$$v_{mn} = \frac{1}{P-1} \sum_{(m',n') \in \mathbb{M}} (g_{m-m',n-n'} - \overline{g}_{mn})^2$$

Die Berechnung lässt sich dabei als Kombination aus Faltungen und dyadischen Punktoperationen darstellen:

$$\mathcal{V} = \mathcal{R}(\mathbb{1} \cdot \mathbb{1}) - (\mathcal{R} \cdot \mathcal{R})$$

Dabei ist $\mathcal{R}$ ein Rechteckfilter, der den Ausmaßen von $\mathbb{M}$ entspricht. Um diese Berechnung noch zu verbessern kann man statt der Rechteck-Filter Binomialfilter verwenden.

Man kann evtl. auch noch Momente höherer Ordnung berücksichtigen, um eine genauere Beschreibung des Grauwerthistogramms zu erhalten:

$$m_{mn}^{(i)} = \frac{1}{P-1} \sum_{(m',n') \in \mathbb{M}} (g_{m-m',n-n'} - \overline{g}_{mn})^i$$

Benutzt man nämlich nur Mittelwert und Histogramm können z.B. die Histogramme in Abb. 12.3 nicht unterschieden werden.

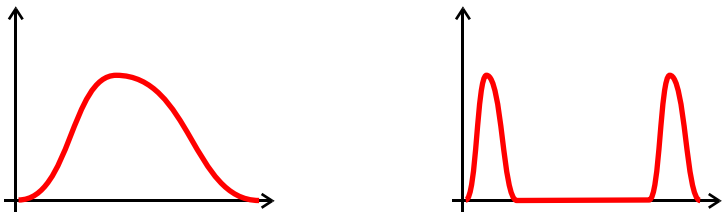


Abb. 12.3: Mit Mittelwert und Varianz ununterscheidbare Histogramme

12.3 Orientierung und Textur

Oft ist es auch wünschenswert die lokale Orientierung einer Textur zu kennen. Wie man diese berechnet wurde ausführlich in Abschnitt 11 über einfache Nachbarschaften beschrieben. Als Parameter für die Textur kann man dann etwa heranziehen:

- Kohärenzmaß der Orientierung: je größer, desto „orientierter“ ist die Textur
- lokale Wellenzahl

12.4 Pyramidale Texturanalyse

Alternativ zur lokalen Wellenzahl kann man mit pyramidalen Strukturen auch die Vergrößerungsabhängigkeit von Texturen untersuchen. So enthält z.B. die Laplace-Pyramide eine Serie von Bandpass-gefilterten Bildern. Man kann also auf den unterschiedlichen Pyramidenstufen unterschiedliche Texturdetails betrachten. Die folgende Abb. 12.4 zeigt verschieden Bandpass-gefilterte Bilder einer Gardine. Man kann leicht sehen, wie unterschiedliche Strukturen in unterschiedlichen Frequenzbändern zum Vorschein kommen.

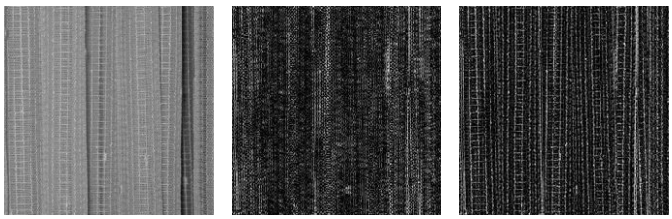


Abb. 12.4: verschieden Bandpass-gefilterte Bilder einer Gardine (Original ganz links)

Mit Laplacepyramiden $\mathcal{L}^{(p)}$ gibt es eine einfache Methode um die Varianz $\mathcal{V}$ zu berechnen:

$$\mathcal{V} = \mathcal{B}(\mathcal{L}^{(p)} \cdot \mathcal{L}^{(p)})$$

Dies entspricht dem üblichen Ausdruck für die Varianz, allerdings ohne den Mittelwert. Da die Laplace-Pyramide nur bandbegrenzte Bilder enthält ist deren Mittelwert (entspricht der überall ausgefilterten Fourierkomponente zu $\vec{k} = 0$) immer 0.

13 Bewegungserkennung

13.1 Grundlagen

In der Bewegungsdetektion geht es darum, festzustellen welche Teile zweier, oder mehrerer Bilder einer Bildsequenz sich geändert haben. Dabei möchte man erfahren, ob sich gewisse Objekte auf den Bildern bewegt haben und gegebenenfalls auch die Richtung und Geschwindigkeit dieser Bewegung feststellen. Man hat zuerst das grundlegende (auf einfache Art unlösbare) Problem, dass Bilder meist eine Projektion einer dreidimensionalen Szene auf eine zweidimensionale Ebene (Kamera) sind. Man kann damit nie die reale Geschwindigkeit $\vec{v} = (v_x, v_y, v_z)^T$ im $\mathbb{R}^3$ ermitteln, sondern nur die Geschwindigkeit $\vec{f} = (f_x, f_y)^T$ auf dem Bild. Diese Geschwindigkeit $\vec{f}$ auf dem Bild wird im folgenden (wie in der Literatur üblich) als **optischer Fluss** bezeichnet. Sie ist die Projektion von $\vec{v}$ auf die Kameraebene. Dies verdeutlicht Abb. 13.1.

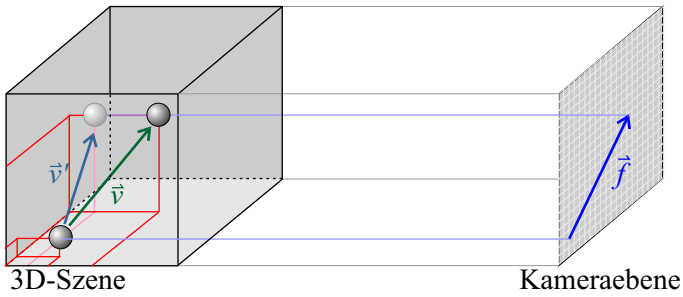


Abb. 13.1: Projektion einer 3D-Bewegung (Geschwindigkeit $\vec{v}$) auf die Kamera-Ebene (optischer Fluss $\vec{f}$). Es zeigt sich, dass verschiedene Bewegungen $\vec{v}$ und $\vec{v}'$ auf den selben optischen Fluss $\vec{f}$ führen.

Ein weiteres grundlegendes Problem, dass hier betrachtet werden soll ist das sog. **Blendenproblem**. Dabei betrachtet man eine Ecke, die sich bewegt (siehe linkes Bild in Abb. 13.2). Solange man die ganze Ecke sieht ist der optische Fluss $\vec{f}$ klar definiert. Betrachtet man aber nur einen Ausschnitt, wie ihn der rechte Teil von Abb. 13.2 zeigt, so kann nur noch eine Normalkomponente $\vec{f}_\perp$ zu der sichtbaren Kante berechnet werden. Anschaulich bedeutet dies, dass man nicht feststellen kann, ob eine Bewegung entlang der Kante (tangential) stattfindet. Dies gilt natürlich nur solange die Kante keine Markierungen aufweist!

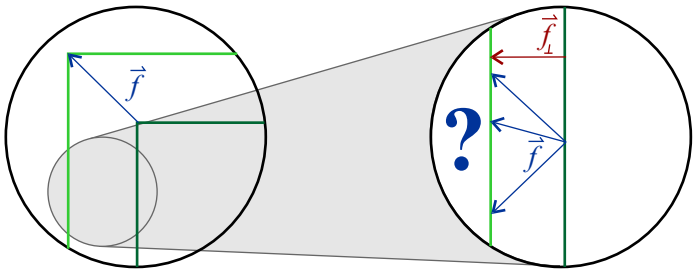


Abb. 13.2: Illustration zum Blendenproblem, nach [Jähne 2005]

Im folgenden wird eine Bildsequenz von Graustufenbildern $g(x, y, t) \equiv g_t(x, y)$ betrachtet.

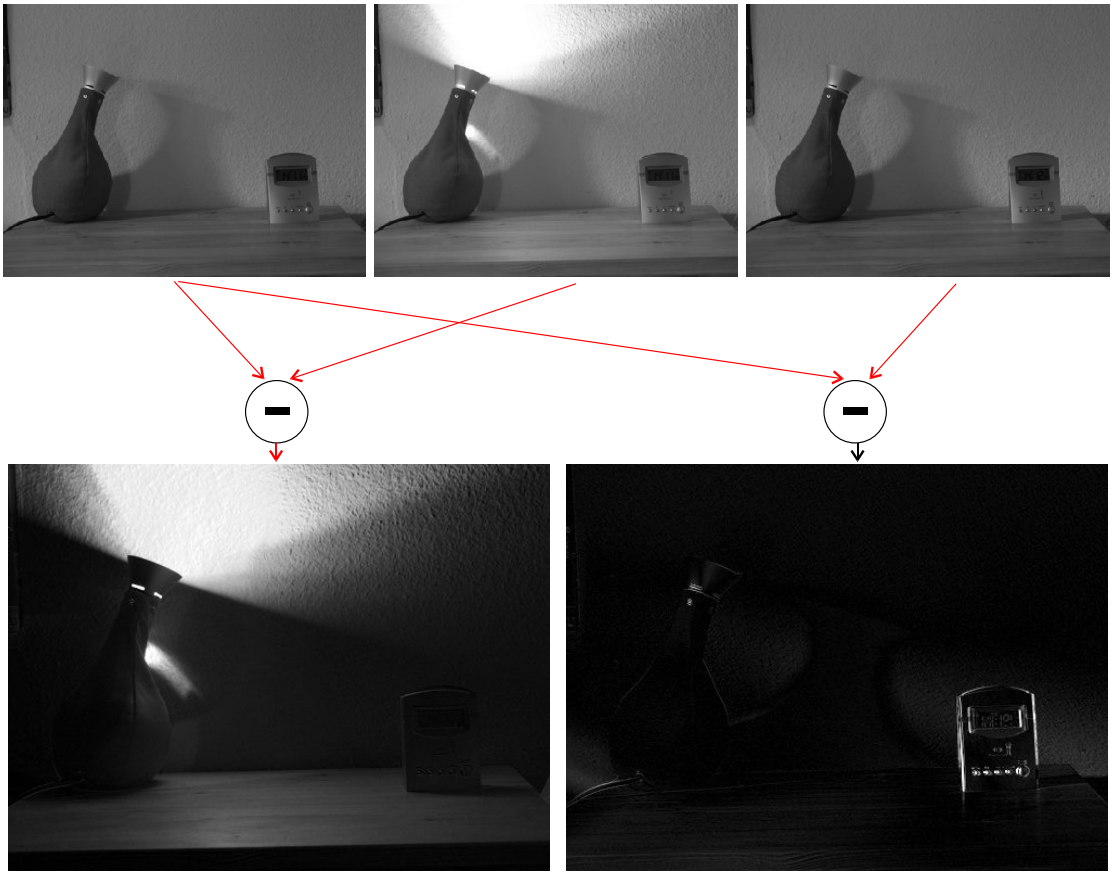
Der optische Fluss gibt die Verschiebung Δx eines Objektes in einer bestimten zeit Δt an, also

$$\vec{f} = \frac{\Delta \vec{x}}{\Delta t}.$$

Damit hat der optische Fluss die Einheit $[\vec{f}] = \frac{\text{Pixel}}{\text{s}}$, wenn man von einem Pixel-Bild ausgeht.

13.2 Einfache Bewegungsdetektion

Differenzenbildung: Eine erste Idee zur Bewegungsdetektion ist es einfach die Differenz zweier Bilder zu betrachten. Hat sich nichts verändert, so sollte die Differenz überall 0, andernfalls erhält man nicht-verschwindende Differenzen. In Abb. 13.3 sind einige Beispiele dargestellt.



Beleuchtungsänderung Bewegung in der Szene

Abb. 13.3: Bewegungsdetektion durch Differenzbildung. Die Differenzbilder sind kontrastverstärkt. Das linke Differenzbild zeigt den Effekt einer Beleuchtungsänderung. Im rechten Bild wurde der Wecker verschoben.

Findet tatsächlich nur Bewegung statt, so kann man dies im Differenzbild $g_\Delta(\vec{x}) = g_{t=0}(\vec{x}) - g_{t=1}(\vec{x})$ sehen. Es zeigt sich also, dass Bewegung in zeitlichen Grauwert-Differenzen resultiert. Mit der hier beschriebenen Technik kann man aber auch nur feststellen, ob es eine Bewegung gegeben hat. Die Ermittlung des optischen Flusses $\vec{f}$ ist nicht so einfach möglich.

Object-Tracking: Eine weitere „einfache“ Methode ist Object-Tracking. Die folgende Abb. 13.4 zeigt die Idee des Verfahrens und die Probleme.

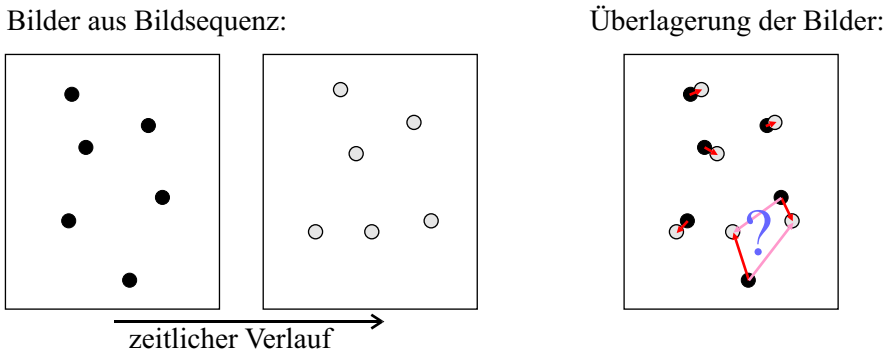


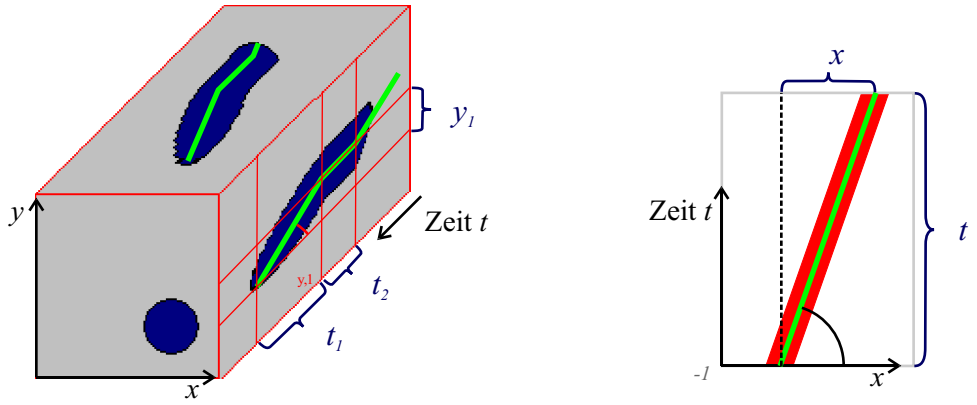
Abb. 13.4: Object-Tracking: Es werden zwei Bilder einer Sequenz verglichen. Rechts sind die Bilder überlagert und die Bewegungsvektoren sind eingezeichnet (optischer Fluss $\vec{f}$). Man erkennt, dass die Methode nicht eindeutig ist (rechts unten!)

Beim Object-Tracking muss man zunächst Objekte in den Bildern einer Zeitserie erkennen (schwarze bzw. graue Kreise in Abb. 13.4). Kann man dann die Objekte i ersten Bild im zweiten Bild wiederfinden, so kann man die Positionsdifferenz berechnen. Diese Differenz $\Delta\vec{x}$ ist ein direktes Maß für den optischen Fluss: $\vec{f} = \frac{\Delta\vec{x}}{\Delta t}$. Dies ist recht einfach möglich, wenn man wenige Objekte hat, die sich nur wenig bewegen. Bei großen Bewegungen ist die Zuordnung u.U. nicht mehr eindeutig, sodass das Verfahren fehlschlägt. Die Korrelationsmethode aus Abschnitt 13.6 ist im Prinzip eine Object-Tracking-Methode.

13.3 Bewegung im Orts-Zeit- und Fourier-Raum

13.3.1 Orts-Zeit-Raum

Die folgende Abb. 13.5 zeigt zwei Bildsequenzen mit einer Bewegung. Man kann leicht erkennen, dass eine Bewegung im Orts-Zeit-Raum ($\{x, t\}$, bzw. $\{x, y, t\}$) eine Linie darstellt. Die Neigung der Linie gibt die Geschwindigkeit (optischer Fluss f, bzw. $\vec{f}$) an. Verläuft die Linie parallel zur zeit-Achse, liegt keine bewegung vor. In der 2D-Sequenz bleibt der Kreis in der Mitte ein für kurze Zeit stehen, um sich dann mit geänderter Geschwindigkeit weiterzubewegen.



2D-Bildsequenz mit verschiedenen Geschwindigkeiten des blauen Kreises

1D-Bildsequenz mit konstanter Geschwindigkeit des roten Objektes

Abb. 13.5: Bewegung in Raum und Zeit für eine 1D und eine 2D Bildsequenz. Die dicke grüne Linie bezeichnet jeweils die Bewegung. Auf den flanken der 2D-Sequenz sind Schnitte durch das Bild aufgezeichnet.

Aus dem 1D-Fall kann man direkt eine Beziehung zwischen Neigung φ und Geschwindigkeit f herstellen:

$$f = \frac{\Delta x}{\Delta t} = -\tan\varphi$$

In der 2D-Sequenz kann man zwei Geschwindigkeiten ablesen:

$$\vec{f} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \begin{pmatrix} \frac{\Delta x}{\Delta t} \\ \frac{\Delta y}{\Delta t} \end{pmatrix} = -\begin{pmatrix} \tan\varphi_x \\ \tan\varphi_y \end{pmatrix}$$

Es zeigt sich also zusammenfassend, dass Bewegung im Orts-Zeit-Raum als Orientierung (über einen Winkel φ gegeben) auftritt. Aus dieser Grunderkenntnis leiten sich die meisten Bewegungserkennungsmethoden her.

13.3.2 Fourier-Raum

Man kann sich nun die obige Bewegung auch im Fourier-Raum betrachten. Der Orts-Zeit-Raum wird von durch die Koordinaten $(\vec{x}, t)$ aufgespannt. Die zugehörigen Wellenzahlen/Frequenzen werden mit $(\vec{k}, \nu)$ bezeichnet. Sie spannen also den Fourier-Raum auf. Man betrachte nun ein Bild, das sich mit konstanter Geschwindigkeit $\vec{u}$ bewegt. Man kann dies so ausdrücken:

$$g(\vec{x}, t) \equiv g(\vec{x} - \vec{u}t).$$

Dies lässt sich nun Fourier-Transformieren:

$$\hat{g}(\vec{k}, \nu) = \vec{F}\big[g(\vec{x}, t)\big] = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} g(\vec{x} - \vec{u}t) \cdot \mathrm{e}^{-2\pi\mathrm{i}\cdot\vec{k}\cdot\vec{x}} \, \mathrm{d}^2x \right] \mathrm{e}^{-2\pi\mathrm{i}\cdot\nu t} \, \mathrm{d}t$$

Nun führt man eine Variablensubstitution durch:

$$\vec{x}' = \vec{x} - \vec{u}t \qquad \Rightarrow \quad \vec{x} = \vec{x}' + \vec{u}t$$

und erhält:

$$\begin{aligned}
\hat{g}(\vec{k}, \nu) &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} g(\vec{x}') \cdot \mathrm{e}^{-2\pi\mathrm{i}\cdot\vec{k}\cdot(\vec{x}'+\vec{u}t)} \, \mathrm{d}^2x' \right] \mathrm{e}^{-2\pi\mathrm{i}\cdot\nu t} \, \mathrm{d}t = \\
&= \int_{-\infty}^{\infty} \underbrace{\left[\int_{-\infty}^{\infty} g(\vec{x}') \cdot \mathrm{e}^{-2\pi\mathrm{i}\cdot\vec{k}\cdot\vec{x}'} \, \mathrm{d}^2x' \right]}_{=\hat{g}(\vec{k})} \mathrm{e}^{-2\pi\mathrm{i}\cdot\vec{k}\cdot\vec{u}t} \cdot \mathrm{e}^{-2\pi\mathrm{i}\cdot\nu t} \, \mathrm{d}t = \\
&= \int_{-\infty}^{\infty} \hat{g}(\vec{k}) \cdot \mathrm{e}^{-2\pi\mathrm{i}\cdot(\vec{k}\cdot\vec{u}+\nu)t} \, \mathrm{d}t = \hat{g}(\vec{k}) \cdot \int_{-\infty}^{\infty} \mathrm{e}^{-2\pi\mathrm{i}\cdot(\vec{k}\cdot\vec{u}+\nu)t} \, \mathrm{d}t = \\
&= \hat{g}(\vec{k}) \cdot \int_{-\infty}^{\infty} \mathrm{e}^{-2\pi\mathrm{i}\omega t} \, \mathrm{d}t = \hat{g}(\vec{k}) \cdot \mathcal{F}_{t,\omega}\Big[1\Big]
\end{aligned}$$

Nun bleibt nur noch ein Integral über die Zeit t übrig, das im letzten Schritt als Fourier-Transformation geschrieben wurde. Es wurde die Substitution $\omega = \vec{k} \cdot \vec{u} + \nu$ eingeführt. Mit dieser schreibt sich das Integral als Fourier-Transformierte vom t-Raum in den ω-Raum. Die Fourier-Trafo von 1 ist $\delta(\omega)$, sodass sich aus obiger Rechnung folgendes Ergebnis ergibt:

$$\Rightarrow \qquad \hat{g}(\vec{k}, \nu) = \hat{g}(\vec{k}) \cdot \delta(\vec{k} \cdot \vec{u} + \nu)$$

Die so erhaltene Fourier-Transformierte $\hat{g}(\vec{k}, \nu)$ ist nur dann ungleich 0, wenn $\vec{k} \cdot \vec{u} + \nu = 0 \;\Leftrightarrow\; \vec{k} \cdot \vec{u} = -\nu$ gilt. Dies ist aber die Definitionsgleichung einer Ebene im Fourier-Raum (Parameter $\vec{u}$, unabhängige Koordinaten: $\vec{k}, \nu$). Dies bedeutet, dass eine konstante Bewegung im Ortsraum im Fourier-Raum auf eine Ebene eingeschränkt bleibt. Diese Situation entspricht der Orientierung bei lokalen Nachbarschaften, nur um eine Dimension erweitert. Hat man diese Ebene ermittelt, so kann man $\vec{u}$ ermitteln:

$$u_x = -\frac{\partial \nu}{\partial k_x} = \underbrace{-\frac{\partial(-k_x u_x - k_y u_y)}{\partial k_x}}_{=u_x}, \qquad u_y \text{ entsprechend}$$

$$\Leftrightarrow \quad \vec{\nabla}_{\vec{k}} \nu = -\vec{u}$$

Dabei bedeutet $\vec{\nabla}_{\vec{k}} = \begin{pmatrix} \partial/\partial k_x \\ \partial/\partial k_y \end{pmatrix}$ der Vektor der ersten Ableitungen nach den $\vec{k}$ (Differentialoperator: Gradient $\vec{\nabla}_{\vec{k}}$). Man erhält also folgendes (zumindest theoretisch anwendbare) Rezept:

1. berechne Fourier-Transformierte der Bildsequenz
2. bestimme die Ebene, die die Bewegung beschreibt
3. berechne $\vec{u}$ aus den Ebenenparametern

Ist das bewegte Objekt ausgedehnt, besteht also nicht nur einem Pixel, so erhält man nicht eine exakte Ebene, sondern eine Art Bande im Fourier-Raum und man muss die am besten passende Ebene darin einpassen (z.B. mit Least-Square-Fit oder Regressionsrechnung).

13.4 Optischer Fluss und Kontinuitätsgleichung

Der optische Fluss wurde als Geschwindigkeit der bewegten Objekte in der Kamera-Ebene eingeführt. Dies ist nur solange richtig, wie sich die Beleuchtung der Szene nicht ändert. Man definiert deswegen:

- **optischer Fluss $\vec{f}$:** sichtbare Bewegung im Bild, also letztendlich Grauwertänderung.
- **Bewegungsfeld $\vec{u}$:** tatsächliche Bewegung des Objektes im Bild, die evtl. nicht sichtbar ist.

Betrachtet man z.B. eine Kugel mit vollkommen ebener Oberfläche, die sich vor einem Hintergrund dreht, so ändert sich am optischen Fluss nichts, also $\vec{f} = 0$ für alle Zeiten t. Trotzdem bewegt sich die Kugel natürlich, sodass $\vec{u} \neq 0$ gilt. Dies ist eben nur nicht messbar. Bringt man auf der Kugel eine Lampe, oder eine andere Markierung an, so wird $\vec{f} \neq 0$ und die Bewegung $\vec{u}$ wird messbar.

Kontinuitätsgleichung: Man kann nun (analog zum Fluss von Flüssigkeiten in der Physik) eine sog. Kontinuitätsgleichung herleiten. Sie lautet:

$$\boxed{\frac{\partial g}{\partial t} + \vec{f} \cdot (\vec{\nabla} g) = \frac{\partial g}{\partial t} + f_x \cdot \frac{\partial g}{\partial x} + f_y \cdot \frac{\partial g}{\partial y} = 0} \tag{13.4.1}$$

Sie stellt einen Zusammenhang zwischen dem optischen Fluss $\vec{f} = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$ und der Grauwertänderung in Raum $\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}$ und Zeit $\frac{\partial g}{\partial t}$ her. Man kann aus ihr versuchen die zwei gesuchten Größen f_x, f_y zu ermitteln.

Anschauliche Erklärung der Kontinuitätsgleichung: In der Physik der strömenden Flüssigkeiten besagt die Kontinuitätsgleichung die Erhaltung der Masse. Die Änderung der Flüssigkeitsmenge an einem Ort kann nur durch Zu- oder Abfluss erfolgen. Für den Grauwert gilt ähnliches. Eine Bewegung stellt einen Zu- oder Abfluss von Grauwert dar. Dabei bedeuten die zwei Terme:

- $\frac{\partial g}{\partial t}$ beschreibt die zeitliche Veränderung des Grauwertes.
- $\vec{f} \cdot (\vec{\nabla} g)$ beschreibt die Änderung des Grauwertes durch einen mit $\vec{f}$ bewegten Grauwertgradienten $\vec{\nabla} g$. Der Gradient tritt hier auf, weil eine Bewegung einer homogenen Fläche ($\vec{\nabla} g = 0$) nicht erkennbar ist. Stehen Gradient und Fluss $\vec{f}$ senkrecht aufeinander, so erfolgt die Bewegung entlang einer Kante des Grauwertes und kann somit nicht detektiert werden, da sich der Grauwert nicht ändert. Sind beide Größen hingegen parallel, so bewegt sich das Objekt entlang seines Gradienten (senkrecht zur Kante) und der Term $\vec{f} \cdot (\vec{\nabla} g)$ ist maximal.

amit bedeutet die Kontinuitätsgleichung

$$\frac{\partial g}{\partial t} = -\vec{f} \cdot (\vec{\nabla} g)$$

nichts anderes, als dass Grauwertänderungen über die Zeit nur von sich bewegenden Objekten/Gradienten ausgelöst werden können.

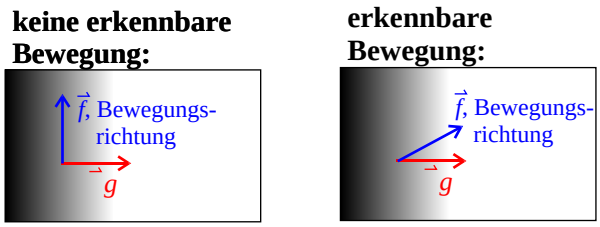


Abb. 13.6: Veranschaulichung zur Kontinuitätsgleichung

1D-Fall: Für 1D-Bildsequenzen kann man f aus dem 1D-Äquivalent zu (13.4.1) bestimmen:

$$\frac{\partial g}{\partial t} + f \cdot \frac{\partial g}{\partial x} = 0 \quad \Rightarrow \quad \boxed{f = -\frac{\frac{\partial g}{\partial t}}{\frac{\partial g}{\partial x}}} \tag{13.4.2}$$

2D-Fall: Da man hier wieder nur eine Gleichung (13.4.1) für zwei Unbekannte $\vec{f} = (f_x, f_y)$ vorliegen hat, ist die Bestimmung von $\vec{f}$ ohne zusätzliche Bedingungen zu stellen nicht möglich. Man kann nur die Normalkomponente des optischen Flusses bestimmen:

$$f_\perp = -\frac{\partial g / \partial t}{\left|\vec{\nabla} g\right|} \tag{13.4.3}$$

Diese Unbestimmtheit entspricht dem in Abschnitt 13.1 geschilderten Blendenproblem.

13.5 Gradientenmethode

Im vorherigen Abschnitt wurde gezeigt, wie man die Kontinuitätsgleichung (13.4.1) nutzen kann um theoretisch den optischen Fluss $\vec{f}$ zu berechnen. Da nur eine Gleichung für zwei Unbekannte vorliegt ist eine eindeutige Lösung nicht immer möglich. Man kann aber eine numerische Optimierungsaufgabe formulieren, die eine Lösung der Kontinuitätsgleichung darstellt:

Man kommt dann zu einem Minimierungsproblem, das den folgenden Ausdruck mit der Methode der kleinsten Quadrate zu minimieren sucht:

$$\|\vec{e}\|_2^2 := \overline{\left(\frac{\partial g}{\partial t} + \vec{f} \cdot \vec{\nabla} g\right)^2} \tag{13.5.1}$$

Die Mittelung $\overline{(...)}$ bedeutet hier auch eine Faltung mit einer Fensterfunktion $w(x, y, t)$. Da man die räumlichen und zeitlichen Gradienten numerisch abschätzen muss, muss man eine Nachbarschaft des betrachteten Punktes $\vec{x}$ mit einbeziehen. Dies wird durch $w(x, y, t)$ erreicht. Man erhält dann:

$$\|\vec{e}\|_2^2 = \int_{-\infty}^{\infty} w(x - x', y - y', t - t') \cdot \left(\frac{\partial g}{\partial t} + \vec{f} \cdot \vec{\nabla} g\right)^2 \, dx \, dy \, dt \tag{13.5.2}$$

Für die Fensterfunktion wählt man bei diskreten Signalen am besten eine Binomialfunktion. Den Ausdruck (13.5.2) minimiert man indem die ersten Ableitungen Null gesetzt werden. Man erhält dann:

$$\begin{aligned} \frac{\partial \|\vec{e}\|_2^2}{\partial f_x} &= \overline{2 \cdot \frac{\partial g}{\partial x} \cdot \left(\frac{\partial g}{\partial t} + \vec{f} \cdot \vec{\nabla} g\right)} \stackrel{!}{=} 0 \\ \frac{\partial \|\vec{e}\|_2^2}{\partial f_y} &= \overline{2 \cdot \frac{\partial g}{\partial y} \cdot \left(\frac{\partial g}{\partial t} + \vec{f} \cdot \vec{\nabla} g\right)} \stackrel{!}{=} 0 \end{aligned}$$

Dies lässt sich einfach als Matrixgleichung auffassen:

$$\mathbf{M} \cdot \vec{f} = \begin{pmatrix} \overline{\partial_x g \cdot \partial_x g} & \overline{\partial_x g \cdot \partial_y g} \\ \overline{\partial_y g \cdot \partial_x g} & \overline{\partial_y g \cdot \partial_y g} \end{pmatrix} \cdot \begin{pmatrix} f_x \\ f_y \end{pmatrix} = - \begin{pmatrix} \overline{\partial_x g \cdot \partial_t g} \\ \overline{\partial_y g \cdot \partial_t g} \end{pmatrix} = \vec{d} \tag{13.5.3}$$

Die Ausdrücke $\overline{\partial_p g \cdot \partial_q g}$ sind Schätzungen für die Produkte der Ableitungen. Sie werden folgendermaßen berechnet:

$$\overline{\partial_p g \cdot \partial_q g} = \mathcal{B}(\mathcal{D}_p \cdot \mathcal{D}_q) g \tag{13.5.4}$$

Dabei ist $\mathcal{B}$ ein Glättungsfilter, der die Fensterfunktion $w(x, y, t)$ implementiert (z.B. eine Binomialmaske). Die Operatoren $\mathcal{D}_q$ ist ein Ableitungsoperator in q-Richtung (z.B. ein Sobel-Operator). Die Operation $\cdot$ entspricht der Punktweisen Multiplikation zweier Bilder.

Das Gleichungssystem (13.5.3) löst man, indem die Matrix M invertiert wird. Dazu berechnet man zuerst ihre Determinante $\det \mathbf{M} = \overline{\partial_x g \cdot \partial_x g} \cdot \overline{\partial_y g \cdot \partial_y g} - \overline{\partial_x g \cdot \partial_y g}^2$ und erhält damit:

$$\vec{f} = \mathbf{M}^{-1} \cdot \vec{d} = \frac{1}{\det \mathbf{M}} \cdot \begin{pmatrix} \overline{\partial_x g \cdot \partial_t g} \, \overline{\partial_y g \cdot \partial_y g} - \overline{\partial_y g \cdot \partial_t g} \, \overline{\partial_x g \cdot \partial_y g} \\ \overline{\partial_y g \cdot \partial_t g} \, \overline{\partial_x g \cdot \partial_x g} - \overline{\partial_x g \cdot \partial_t g} \, \overline{\partial_x g \cdot \partial_y g} \end{pmatrix} \tag{13.5.5}$$

Dies funktioniert natürlich nur solange, wie $\det \mathbf{M} \neq 0$ gilt. Ist dies nicht erfüllt, so liefert das Verfahren keine Schätzung für $\vec{f}$.

13.5.1 Tensormethode

Die Tensormethode verwendet ein etwas anderes Optimierungsverfahren, um den optischen Fluss zu schätzen. Man nimmt an, dass der optische Fluss in Richtung $\vec{e}$ verläuft (mit $\|\vec{e}\| = 1$). Nun ist dieses $\vec{e}$ dann optimal geschätzt, wenn es senkrecht auf dem raum-zeitlichen Gradienten $\vec{\nabla}_{xyt} g$ des Graustufenbildes steht. Diese letzte Bedingung lässt sich über ein euklidisches Skalarprodukt schreiben:

$$\left(\vec{\nabla}_{xyt} g^T \vec{e}\right)^2 \stackrel{!}{=} 0 \tag{13.5.6}$$

Hier wird dann also (analog zum vorherigen Abschnitt) der folgende Ausdruck minimiert:

$$\overline{\left(\vec{\nabla}_{xyt} g^T \vec{e}\right)^2} = \int w(x-x', y-y', t-t') \cdot \left(\vec{\nabla}_{xyt} g^T \vec{e}\right)^2 \, dx\, dy\, dt \rightarrow \text{Minimum} \tag{13.5.7}$$

Dabei wurde wieder eine Fensterfunktion $w(x, y, t)$ eingeführt, die wie vorher gewählt wird. Es zeigt sich, dass sich dieses Problem auf das Auffinden des Eigenvektors zum kleinsten Eigenwert des sog. *Strukturtensors* **J** (daher der Name des Verfahrens) reduziert. Dieser hat die folgende Form:

$$\mathbf{J} = \begin{pmatrix} \overline{\partial_x g \partial_x g} & \overline{\partial_x g \partial_y g} & \overline{\partial_x g \partial_t g} \\ \overline{\partial_x g \partial_y g} & \overline{\partial_y g \partial_y g} & \overline{\partial_y g \partial_t g} \\ \overline{\partial_x g \partial_t g} & \overline{\partial_y g \partial_t g} & \overline{\partial_t g \partial_t g} \end{pmatrix} \tag{13.5.8}$$

Dabei sind die Mittelwerte $\overline{(...)}$ genauso wie im vorherigen Abschnitt definiert. Mit numerischen Methoden kann man nun die drei Eigenwerte (und die zugehörigen Eigenvektoren) von **J** ermitteln. Sie seien danach mit $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ in absteigender Reihenfolge gegeben. Die Eigenvektoren sind dann $\vec{e}_1, ..., \vec{e}_3$. Man kann aus den drei Eigenwerten nun einige verschiedene Bewegungstypen ableiten:

- **Keine Bewegung** ($\lambda_1 = \lambda_2 = \lambda_3 = 0$)**:** Diese Bedingung kann auch einfacher überprüft werden (ohne die Eigenwerte zu berechnen): Dazu berechnet man einfach die Spur von **J**, da **J** nach einer Hauptachsentransformation gerade Diagonalgestalt hat, mit den Eigenwerten auf der Diagonalen und die Spur unter einer solchen Transformation invariant ist.
- **Blendenproblem** ($\lambda_1 > 0, \lambda_2 = \lambda_3 = 0$)**:** Hier tritt das Blendenproblem auf und es kann nur die Normalkomponente des optischen Flusses berechnet werden:

$$\vec{f}_\perp = -\frac{e_{1t}}{e_{1x}^2 + e_{1y}^2} \begin{pmatrix} e_{1x} \\ e_{1y} \end{pmatrix}, \qquad \left\|\vec{f}_\perp\right\| = \sqrt{\frac{e_{1t}^2}{1 - e_{1t}^2}}. \tag{13.5.9}$$

- **konstante Bewegung** ($\lambda_1, \lambda_2 > 0, \lambda_3 = 0$)**:** Hier lässt sich der optische Fluss schätzen:

$$\vec{f} = -\frac{1}{e_{3t}} \begin{pmatrix} e_{3x} \\ e_{3y} \end{pmatrix}, \qquad \left\|\vec{f}\right\| = \sqrt{\frac{e_{3t}^2}{1 - e_{3t}^2}}. \tag{13.5.10}$$

- **nichtkonstante Bewegung** ($\lambda_1, \lambda_2, \lambda_3 > 0$)**:** Hier kann keine sinnvolle Schätzung des optischen Flusses gegeben werden.

Die numerische Berechnung der Eigenwerte und -vektoren lässt sich mit Standard-Verfahren (z.B. Givens-Householder + QL-Zerlegung) durchführen. Mit Hilfe der Symmetrie des Tensors **J** lässt sich aber das Eigenwertproblem auch analytisch lösen, was in [Kahder etal. 2001] durchgeführt wird. Damit erhält man eine Beschleunigung der Berechnung.

13.6 Korrelationsmethode

Die Korrelationsmethode arbeitet nach dem Object-Tracking-Prinzip. Es werden nur zwei Bilder einer Zeitserie verglichen. Sie seien hier mit $g_1 \equiv g_t$ und $g_2 \equiv g_{t+\Delta t}$ bezeichnet. Die Korrelationsmethode berechnet die Verschiebung $\vec{s}$ eines Objektes in den Bildern in der zeit Δt. Es stellt sich also zunächst die Frage, wann man davon sprechen kann, dass zwei Objekte in den Bildern g_1 und g_2 gleich sind. Es ist wünschenswert wenn man Beleuchtungsänderungen ignorieren kann, also sollen zwei Objekte gleich sein, wenn sich ihr Grauwert bis auf einen Faktor α (Beleuchtung) gleicht. Dies entspricht zwei parallelen (aber unterschiedlich langen) Merkmalsvektoren.

Um dieses Prinzip zu implementieren wird der sog. **Kreuzkorrelationskoeffizient** $r(\vec{s})$ maximiert:

$$r(\vec{s}) = \frac{\int\limits_{-\infty}^{\infty} g_1(\vec{x}) \cdot g_2(\vec{x}-\vec{s})\,\mathrm{d}^2x}{\sqrt{\int\limits_{-\infty}^{\infty} g_1(\vec{x})\,\mathrm{d}^2x \cdot \int\limits_{-\infty}^{\infty} g_2(\vec{x}-\vec{s})\,\mathrm{d}^2x}} = \frac{\overline{g_1(\vec{x}) \cdot g_2(\vec{x}-\vec{s})}}{\left(\overline{g_1^2(\vec{x})} \cdot \overline{g_2^2(\vec{x}-\vec{s})}\right)^{1/2}}$$

Dieser Koeffizient $r(\vec{s})$ hängt von der Verschiebung $\vec{s}$ ab und ist ein Maß für die Ähnlichkeit der zwei Bilder g_1 und g_2. Er ist 0 wenn die Bilder vollkommen unähnlich sind und 1 wenn sie sich exakt entsprechen. Man berechnet nun r in Abhängigkeit von der Verschiebung. Bei der richtigen Verschiebung sind sich die Bilder gleich und man hat ein Maximum von $r(\vec{s})$ gefunden. Der Nenner von $r(\vec{s})$ sorgt für die Normierung und die Unabhängigkeit von der Beleuchtung. In der Integration wird man im realen Algorithmus noch eine Fensterfunktion $w(\vec{x})$ berücksichtigen. Sie wird damit dann zu einer Mittelung oder Glättung (z.B. Binomial/Gauß-Maske $w(\vec{x})$). Man bestimmt dann die Verschiebung $\Delta\vec{x}$ und den optischen Fluss $\vec{f}$ so:

$$\Delta\vec{x} = \max_{\vec{s}} r(\vec{s}) \qquad \Rightarrow \qquad \vec{f} = \frac{\Delta\vec{x}}{\Delta t}$$

Die Maximierung kann z.B. mit dem Newton-Raphson-Verfahren berechnet werden.

Evaluation: Das Verfahren ist (nach Konstruktion) unempfindlich gegenüber Beleuchtungswechseln, was es von den Verfahren, die auf der Kontinuitätsgleichung basieren abhebt. Dafür ist der Rechenaufwand für das Auffinden des Maximums sehr hoch. Außerdem basiert das Verfahren auf dem Vergleich zweier Bilder und kann nicht auf das Kontinuum erweitert werden. Es ist damit z.B. empfindlich auf kleine Bildschwankungen, die bei einer Betrachtung einer ganzen Zeitserie herausgefiltert werden könnten.

13.7 Phasenmethode

Die Phasenmethode ist ebenfalls ein Verfahren, das versucht den Einfluss schwankender Beleuchtung zu minimieren. Dazu wählt es aber einen anderen Ansatz, wie die Korrelationsmethode. Man wählt hier nicht den Grauwert als Ausgangsdaten, sondern die lokale Phase (siehe auch Abschnitt 11.4) des Bildes. Um das zu verstehen betrachtet man zunächst eine einfache ebene Welle in Raum und Zeit:

$$g(x,t) = g_0 \cdot \cos\big[2\pi(kx - \nu t\big] = g_0 \cdot \cos\big[2\pi(kx - ukt\big]$$

Dabei ist ν die Frequenz der Welle und $u = \nu/k$ ihre Fortbewegungsgeschwindigkeit. k ist der Wellenvektor der Schwingung. Die Phase ist das Argument des Cosinus, also:

$$\phi(x,t) = 2\pi(kx - ukt)$$

Die Geschwindigkeit u der Welle erhält man durch Differenzieren der lokalen Phase:

$$\frac{\partial\phi}{\partial x} = 2\pi k \qquad \frac{\partial\phi}{\partial t} = -2\pi\nu \qquad \Rightarrow \qquad u = \frac{\nu}{k} = -\frac{\partial\phi/\partial t}{\partial\phi/\partial x}$$

Man kann also über die Bestimmung der Phase ϕ die Bewegungsgeschwindigkeit u von Objekten im Bild bestimmen. Die Phase wird (wie in 11.4 beschrieben) über Hilbert- oder Quadraturfilter berechnet.

Evaluation: Da die Phase von der Beleuchtung unabhängig ist (diese ändert nur die Amplitude g_0) ist das verfahren nahezu unabhängig von der Beleuchtung. Dafür ist diese Methode eigentlich nur für 1D-Sequenzen geeignet. Man kann dies umgehen, indem man vorher eine Richtungszerlegung des Bildes vornimmt. Außerdem hat man die üblichen Probleme der Quadraturfilter. Vor der Bestimmung der Bewegung muss man das Bild mit einem Bandpass filtern. Es zeigt sich, dass die Richtungszerlegung auf Laplace-Pyramiden eine gute Ausgangssituation für diese Methode ist.

14 Inverse Filterung

Viele physikalische Bildaufnahmesysteme sind „beugungsbegrenzt“. Die Auflösung ist dabei als der kleinste Abstand zweier Punkte definiert, die noch getrennt wahrgenommen werden. Dies bedeutet, dass die Auflösung eines Mikroskops, Teleskops oder Kamerobjektivs von seiner Öffnungsweite abhängt. Die übliche Punktantwort eines optischen Systems ist ein sinc- bzw. Airy-Funktion. Sie führt zu einer Verschmierung kleiner Objekte und zu Ringen z.B. um Sterne im Teleskop. Man nimmt hier also an, dass man die Störung des Bildes durch das optische System als Faltung des realen Bildes g_r mit einem Filter h für das Aufnahmesystem modellieren kann. Es stellt sich nun die Frage, ob es möglich ist diese Bildstörung rückgängig zu machen.

Im Prinzip muss durch eine Filterung keine Information verloren gehen. Angenommen das gemessene Bild lässt sich als Faltung darstellen

$$g = h \circledast g_r,$$

so entspricht dies im Fourier-Raum einer Multiplikation:

$$\hat{g} = \hat{h} \cdot \hat{g}_r,$$

Es spricht nun nichts dagegen diese Multiplikation wieder rückgängig zu machen. Dies bezeichnet man als Rückfaltung:

$$\hat{g}_r = \frac{\hat{g}}{\hat{h}} = \hat{h}^{-1} \cdot \hat{g},$$

Man kann also das reale Bild im Fourierraum berechnen und dann in den Realraum zurücktransformieren:

$$g_r = \mathcal{F}^{-1}\left[\frac{\mathcal{F}[g]}{\mathcal{F}[h]}\right]$$

Die Rückfaltung durch Division im Fourierraum führt nur dann zu Problemen, wenn $\hat{h}$ Nullstellen hat. Liegt dieser Fall vor, so ist durch die Faltung Information vernichtet worden, die nicht wieder hergestellt werden kann. Als Beispiel kann man etwa die Glättung durch ein 3×3-Binomialfilter betrachten. Wenn das Bild 100×100 Pixel groß ist, so sind nur wenige Pixel von $\hat{h}$ von null verschieden, sodass eine Rückfaltung nicht möglich ist. Dies zeigt deutlich, dass eine Glättung Information vernichtet.

Zusätzlich bekommt man Probleme mit Rauschen. Wenn das Rauschen n nach der Faltung aufaddiert wird, so erhält man:

$$g = h \circledast g_r + n \quad \circ\!\!-\!\!\bullet \quad \hat{g} = \hat{h} \cdot \hat{g}_r + \hat{n} \;\Rightarrow\; \hat{g}_r = \frac{\hat{g}}{\hat{h}} - \frac{\hat{n}}{\hat{h}}$$

Damit wird das Rauschen in den Bereichen verstärkt, in denen $\hat{h}$ klein ist. Diese große Verstärkung des Bildrauschens macht obigen Rückfaltungs-Ansatz sehr schlecht.

Man wählt daher iterative Ansätze, um die Faltung rückgängig zu machen. Eine einfache Möglichkeit ist der folgende Ansatz für die Umkehrung eines Unschärfeoperators $\mathcal{H}$ mit Transferfunktion $\hat{h}$:

$$\hat{h}^{-1} = \frac{1}{\hat{h}} = \frac{1}{1 - \hat{h}'} \quad \text{mit } \hat{h}' = 1 - \hat{h}$$

Diesen Ausdruck für $\hat{h}'$ kann man nach Taylor entwickeln:

$$\hat{h}^{-1} = 1 + \hat{h}' + \hat{h}'^2 + \hat{h}'^3 + ...$$

Van-Critter-Iteration:

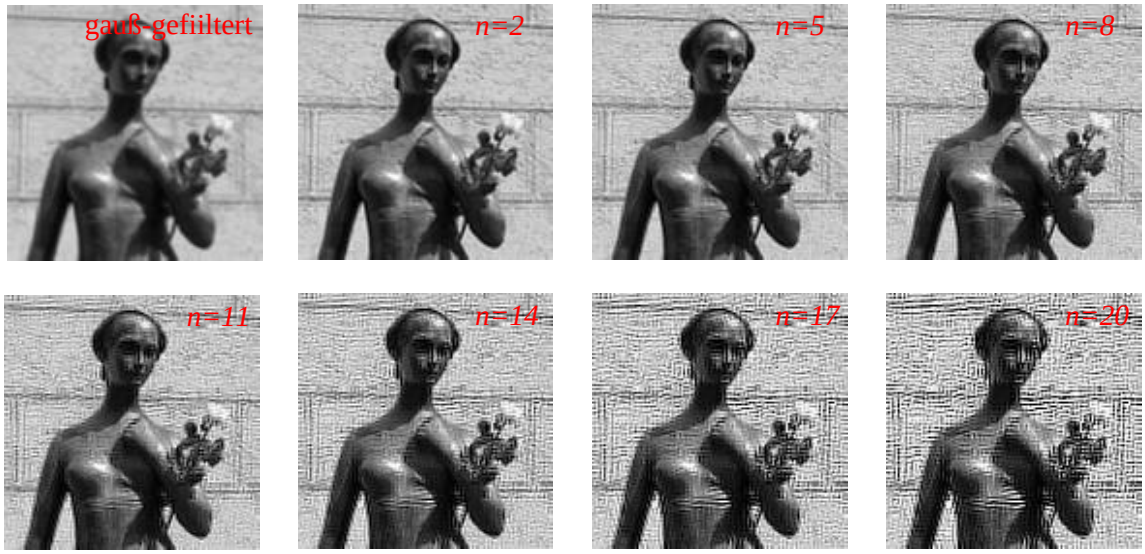


Abb. 14.1: Van-Critter-Iteration auf einem weich-gezeichneten Testbild

Im Realraum entspricht dies der folgenden Reihe:

$$\mathcal{H}^{-1} = \mathcal{I} + \mathcal{H}' + \mathcal{H}'^2 + \mathcal{H}'^3 + \ldots \quad \text{mit} \quad \mathcal{H}' = \mathcal{I} - \mathcal{H}$$

Man kann also einen iterativen Prozess angeben, der immer mehr Terme hinzunimmt und damit die Rückfaltung von Schritt zu Schritt verbessert. Diesen Prozess kann man dann abbrechen, wenn das Rauschen sichtbar wird. Die einzige Voraussetzung ist, dass die Verschmierung $\mathcal{H}$ bekannt ist. Diese lässt sich aber oft mit Testbildern vermessen (z.B. als Bild einer Punkt-Lichtquelle, wie etwa einem Stern oder einem sehr kleinen beleuchteten Staubkorn). Die direkte Implementierung dieses Schemas ist aber nicht sinnvoll, da im i-ten Schritt mit der Maske $\mathcal{H}^i$ gefaltet werden muss, die aber von Schritt zu Schritt immer größer wird. Man verwendet deswegen das sog. Horner-Schema, um das obige Polynom zu berechnen:

$$g_0 = g, \qquad g_{k+1} = g + \mathcal{H}' g_k$$

Daraus erhält man:

$$\begin{aligned} g_0 &= g \\ g_1 &= g + \mathcal{H}' g \\ g_2 &= g + \mathcal{H}' g + \mathcal{H}'^2 g \\ &\vdots \end{aligned}$$

Dieses Iteration wird **Van Critter Iteration** genannt. In Abb. 14.1 ist ein Beispiel der Anwendung gezeigt. Man sieht, dass zu Anfang das Bild immer schärfer wird, während es nach zu vielen Iterationen nur noch aus Störungen besteht.

Teil IV

Bildanalyse

15 Segmentierung

15.1 Einleitung

Die Segmentierung bestimmt, welche Bereiche eines Bildes zu einem Objekt gehören. Die so entstandenen Regionenbilder können dann weiterverarbeitet, oder zur Klassifizierung herangezogen werden. Es entsteht ein **Binärbild**, d.h. jeder Pixel kann nur die Werte 0 oder 1 annehmen:

$$g_{mn} = \begin{cases} 1: & \text{Pixel gehört zu einem Objekt} \\ 0: & \text{Pixel gehört nicht zu einem Objekt} \end{cases}$$

Es gibt verschiedene Segmentierungsmethoden, die in den folgenden Abschnitten behandelt werden:

- **pixelbasierte Methoden** schenken der lokalen Nachbarschaft keine Beachtung
- **kantenbasierte Methoden** nutzen nur Diskontinuitäten (Kanten) im Bild
- **regionenbasierte Methoden** nutzen homogene Bildbereiche, um Objekte aufzufinden
- **modellbasierte Methoden** nutzen Informationen über die bekannte Form der zu segmentierenden Objekte

Die einfachsten Methoden basieren dabei nur auf lokaler Information (kleine Nachbarschaft).

15.2 Pixelorientierte Methoden

Die pixelorientierten Segmentierungsmethoden sind die einfachste Klasse von Segmentierungsalgorithmen. Sie klassifizieren Bildpunkte anhand eines Schwellwertes für ihren Grauwert. Die folgende Abb. 15.1 zeigt zwei Bilder mit Objekten, die klassifiziert werden sollen. Dazu sind jeweils die Histogramme gezeichnet. Diese zeigen einen starken bimodalen Charakter, d.h. es existieren zwei extreme Maxima, zwischen denen nur wenige Pixel zu finden sind. Ein Maximum gehört zu den Objekten; das zweite zum Hintergrund.

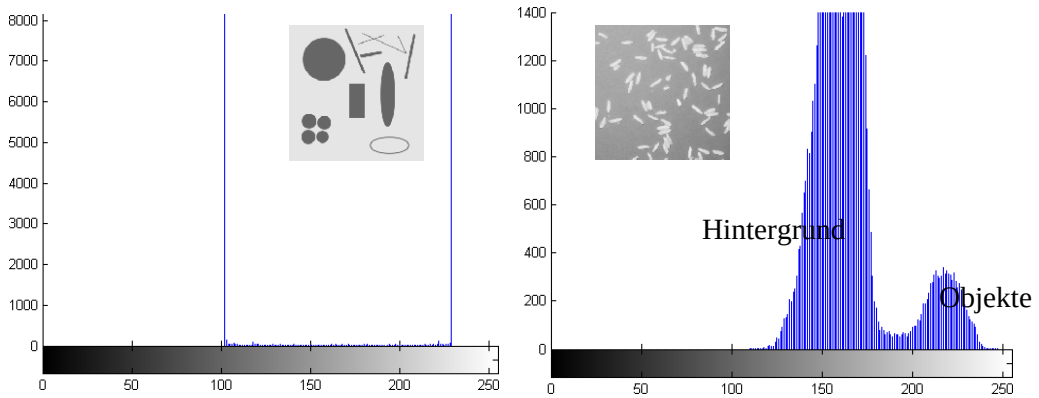


Abb. 15.1: bimodale Histogramme zweier Bilder

Bei solchen Bildern mit bimodalen Histogrammen ist die pixelorientierte Segmentierung gut geeignet. Sie geht so vor:

1. suche einen Schwellwert s, mit dem das Bild segmentiert werden kann
2. überprüfe jeden Pixel g_{mn}
   a) Wenn $g_{mn} < s$, so gehört der Pixel zum Hintergrund
   b) Wenn $g_{mn} \geq s$, so gehört der Pixel zum Objekt

Evaluierung: Die folgende Abb. 15.2 zeigt, welchen Einfluss die Wahl des Schwellwertes auf das Ergebnis hat.

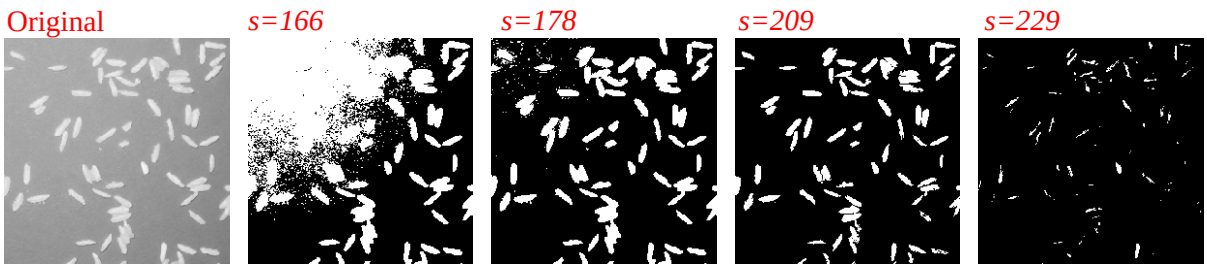


Abb. 15.2: Segmentierung eines Bildes mit dem Schwellenwertverfahren mit verschiedenen Schwellenwerten s.

Im Bild $s = 166$ sieht man Probleme, die man mit inhomogener Beleuchtung haben kann. Rechts unten werden die Reiskörner noch gut segmentiert, während links oben der Hintergrund als Objekt aufgefasst wird. Im Bild $s = 178$ sieht man links oben einige Fehlpixel, die durch Rauschen entstehen können. Außerdem sind die Reiskörner rechts unten zu breit. Das Bild $s = 209$ ist ein ziemlich guter Kompromiss. Im Bild $s = 229$ fehlen einige Reiskörner, weil sie zu dunkel sind; andere sind zu klein.

Fehler: Zusammenfassend findet man folgende Fehler:

- inhomogene Beleuchtung führt evtl. zu verfälschten Ergebnissen (der Hintergrund wird als Objekt klassifiziert.
- sind helle und dunkle Objekte im Bild enthalten, so werden bei einem globalen Schwellenwert helle Objekte eher zu groß segmentiert und dunkle Objekte eher zu klein. Abb. 15.3 verdeutlicht das.
- Durch Rauschen können einzelne Störpixel entstehen.

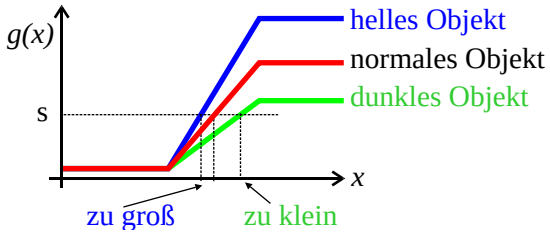


Abb. 15.3: Größen-Fehler beim Segmentieren bei verschieden hellen Objekten

Festlegen des Schwellwertes: Um den Schwellwert richtig festzulegen ist es nötig die Kanten der Objekte im Bild zu betrachten. Es können lineare oder nicht-lineare Kanten vorliegen (siehe Abb. 15.4). Ist die Kante ein linearer Anstieg/Abfall, so kann der Schwellwert einfach auf die halben Kantenhöhe gesetzt werden. Bei nicht-linearen Kanten verschiebt er sich. Würde man ihn hier anders festlegen, so

wären alle Objekte eher zu groß (siehe die graue Linie in der rechten Abbildung). Die ideale Segmentierungsgrenze liegt auf der halben Kantenbreite. Unter Umständen können solche Fehler von nicht-linearen Kanten aber auch vernachlässigt werden (je nach Anwendung!).

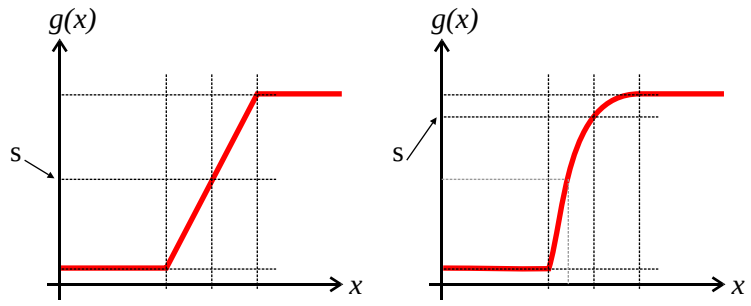


Abb. 15.4: Festlegung des Schwellenwertes s bei linearen und nicht-linearen Kanten

15.3 Kantenbasierte Segmentierung

Dieses verfahren vermeidet den Größenfehler der Schwellenwertmethode. Ein solcher Algorithmus läuft so ab:

1. Berechne den Betrag des Gradienten des Bildes $\left|\vec{\nabla} g\right|$, als Maß für die Kantenstärke
2. Suche (zeilenweise) ein Maximum in $\left|\vec{\nabla} g\right|$
3. Folge der zugehörigen Kante, bis der Anfangspunkt wieder erreicht ist
4. suche das nächste Maximum und fahre mit Schritt 3 fort, bis alle Pixel auf ein Maximum untersucht wurden

Dieser Algorithmus setzt voraus, dass es sich bei den Objekten um zusammenhängende Gebiete handelt. Die beiden Objekte in Abb. 15.5 können deswegen u.U. nicht unterschieden werden und führen beide zum selben segmentierten Gebiet.

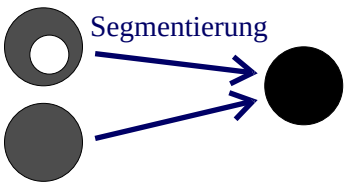


Abb. 15.5: Ununterscheidbare Objekte beim kantenbasierten Segmentieren

Fehleranalyse bei inhomogener Beleuchtung: Man kann leicht zeigen, dass bei der kantenbasierten Segmentierung kein Größenfehler entsteht, wenn die Beleuchtung maximal linear variiert. Dazu betrachtet man ein einfaches Bild mit einer steil ansteigenden Kante.

Die Kante soll inhomogen (ortsabhängig) beleuchtet werden. Die Beleuchtung wird als Parabel modelliert (linear mit $b_2 = 0$ und gleichmäßig mit $b_1 = 0$):

$$b(x) = b_0 + b_1 x + b_2 x^2$$

Die Kante wird zusätzlich durch eine Filtermaske $h(x)$ verschmiert (Faltung). Die folgende Abb. 15.6 zeigt diese Schritte.

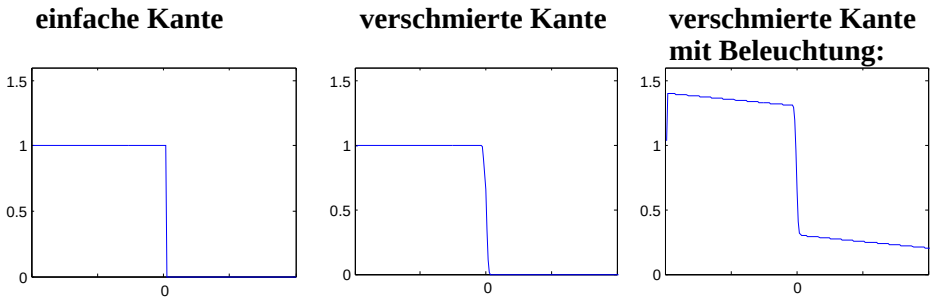


Abb. 15.6: Modellierung einer Kante im Bild mit Verschmierung und Beleuchtung

Man kann das Bild nach Verschmierung und Beleuchtung so schreiben (Die Kantenform wurde in den Integralgrenzen berücksichtigt):

$$g(x) = g_0 \underbrace{\int_{-\infty}^{x} h(x)\,\mathrm{d}x}_{\text{Faltung}} + \underbrace{b_0 + b_1 x + b_2 x^2}_{Beleuchtung}$$

Nun kann man auch das Gradientenbild anschreiben:

$$\frac{\partial g}{\partial x} = g_0 \cdot h(x) + b_1 + 2b_2 x$$

Man nimmt nun an, dass sich $h(x)$ um $x = 0$ in etwa wie eine Parabel verhält, also $h(x) \approx h_0 - h_2 x^2$. Dies kann man in obiges Gradientenbild einsetzen.

$$\frac{\partial g}{\partial x} \approx g_0 \cdot (h_0 - h_2 x^2) + b_1 + 2b_2 x$$

Möchte man nun die Lage der Maxima dieses Bildes bestimmen, so bildet man dessen Ableitung und setzt diese null:

$$\frac{\partial^2}{\partial x^2}\frac{\partial^2 g}{\partial x^2} = -2g_0 h_2 x + 2b_2 = 0$$

Daraus erhält man:

$$x_{\text{kante}} = \frac{b_2}{g_0 h_2}$$

Das Ergebnis $x_{\text{kante}} = 0$ wäre ideal. Aus obiger Gleichung kann man folgende Schlussfolgerungen ziehen:

1. bei maximal linearer Beleuchtung ($b_2 = 0$) wird die Kante immer an der richtigen Position erkannt. Die Position hängt dann auch nicht von der Objekthelligkeit ab.
2. bei nicht-linearer Hintergrundbeleuchtung steigt der Fehler mit der Verschmiertheit der Kante (entspricht h_2) und der Objekthelligkeit g_0 an.

15.4 Regionenorientierte Verfahren

Die Standardverfahren hier sind Region-Growing und Split-and-Merge. Bei ersterem erweitert man das Gebiet um einen Saatpunkt solange, bis man an eine Kante stößt. das Split-and-merge-verfahren teilt eine Fläche solange in kleine Unterflächen auf, bis eine Unterfläche nahezu homogen ist. Sie kann kann dann als Objekt oder Hintergrund mit einem einfachen Schwellenwert klassifiziert werden.

Pyramid-Linking ist ein iteratives, regionenorientiertes Segmentierungsverfahren. Es geht vom Grauwertbild aus, nicht vom Merkmalsbild, denn das verfahren berechnet das Merkmal selber. Angenommen das zu segmentierende Merkmal wird durch ein Filter $\mathcal{M}$ berechnet. Stößt dieses Filter an die Grenzen eines Objekte, so wird sein Wert verfälscht, da auch Hintergrundpixel eine Rolle spielen. Am besten wäre es hier die Maske $\mathcal{M}$ auf das Objekt zu beschränken. Dies ist aber vor der Segmentierung nicht möglich, während aber gleichzeitig das Merkmalsbild zur Segmentierung benötigt wird. Man hat also ein klassisches Henne-und-Ei-Problem. Man verwendet zu seine Lösung ein iteratives Verfahren. Beim Pyramid-Linking ist $\mathcal{M}$ einfach ein Mittelungsfilter. Das verfahren läuft folgendermaßen ab:

1. **Gauß-Pyramide:** Berechne die Gauß-Pyramide $G^{(i)}$ des Bildes g. Dabei wird über jeweils vier Pixel gemittelt. Damit trägt jeder Bildpunkt zu zwei Nachbarpunkten bei.
2. **Segmentierung:** Jeder Bildpunkt trägt zu zwei Knoten auf der nächsten Ebene der Gauß-Pyramide bei. Man entscheidet nun für jeden Punkt zu welchem der beiden er eher gehört (an welchem er näher liegt) und verbindet diese. Dies wird nun für alle Pixel aller Ebenen durchgeführt. Man erhält so einen Baum, der aus den Pixeln der Gauß-Pyramide besteht.
3. **Mittelung miteinander verbundener Pixel:** Nun mittelt man neu, indem nur noch die mit einem Elternknoten verbundenen Kindpixel zur Mittelung herangezogen werden. Nun kehrt man wieder zu Schritt 2 zurück und wiederholt dies, bis sich die Verbindungen nicht mehr ändern.
4. Im Ergebnisbaum kann man nun die verschiedenen Regionen identifizieren. Jede Region stellt einen eigene Teilbaum dar.

Das Verfahren ist in Abb. 15.7 illustriert. Die Anzahl der Segmentierungsebenen ergibt sich aus der Anzahl der Teilbäume in den ersten Schichten nach der Wurzel des Baumes und muss nicht unbedingt vorher bekannt sein!

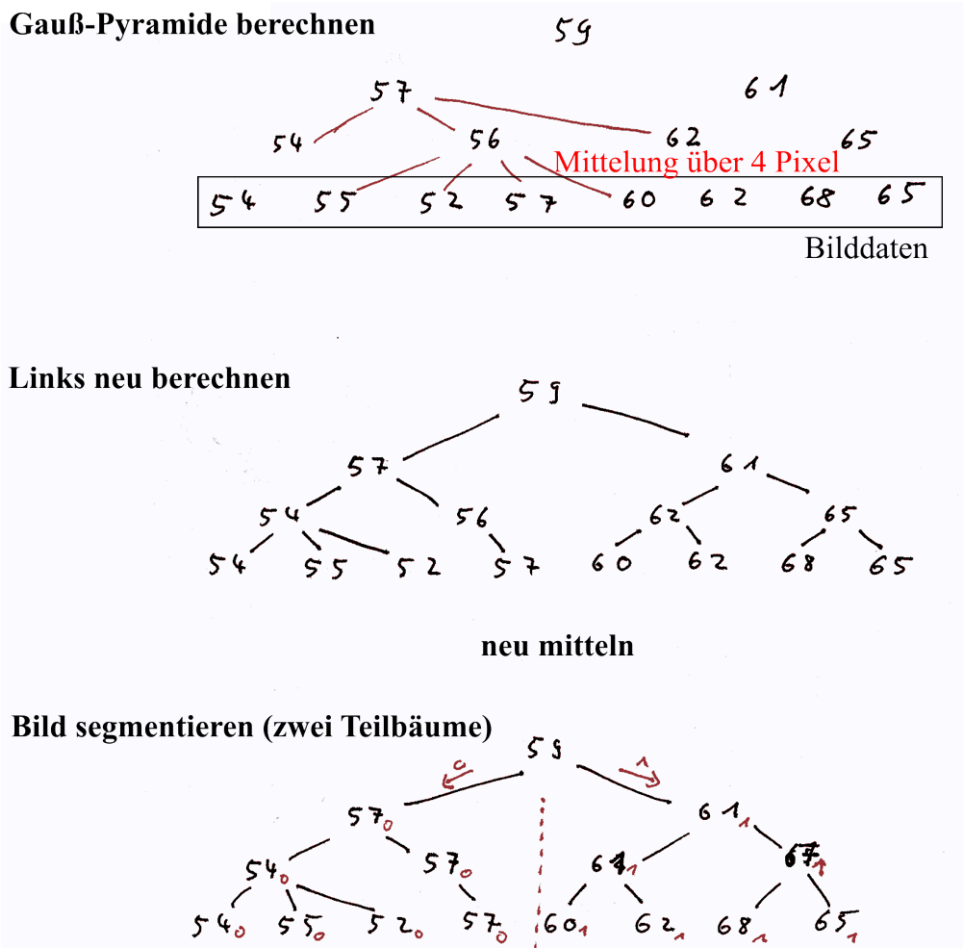


Abb. 15.7: Veranschaulichung des pyramid-linking-Verfahrens

15.5 Modellbasierte Segmentierung

15.5.1 Regularisierung und Modellierung

Die bisher besprochenen verfahren verwenden keine Information über die Form der zu segmentierenden Objekte. Die folgenden Verfahren machen Annahmen über die Form enthaltenen Objekte und nutzen diese zusätzliche (globale) Information zur Segmentierung. Man erstellt also ein Modell der Wirklichkeit und versucht damit die Segmentierung zu verbessern.

Modell: Beim Aufstellen eines Modells muss man sich einiger Tatsachen bewusst sein. Es ist wichtig, dass Modelle die Wirklichkeit immer nur bis zu einem gewissen Grad wiederspiegeln können. Man muss sich also auch Gedanken über die Grenzen des Modells machen. So wird man etwa mit der Modellierung von Objekten als Rechtecke keine Kreise segmentieren können. Außerdem muss ein Modell auch dann nicht die Wirklichkeit beschreiben, wenn es perfekt zu den Daten passt. In Abb. 15.8 ist ein Beispiel gezeigt. Das Modell nimmt ein einziges schwarzes Objekt auf weißen Untergrund an. Das Bild rechts führt aber zum gleichen Grauwerthistogramm. Damit würde das rechte Bild gleich klassifiziert, wie das linke und das Modell ist leider falsch, obwohl es zu den Daten passt.

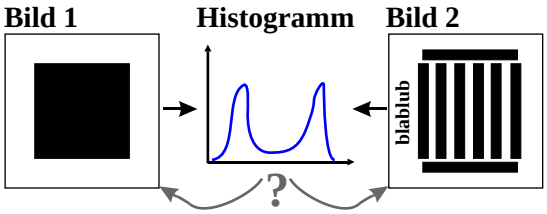


Abb. 15.8: falsches Modell (Bild 1) für die Daten (Bild 2)

15.5.2 Hough-Transformation

Dieses verfahren erkennt gerade Kanten. Dazu geht man davon aus, dass jeder Punkt einer Kante auf einer Geraden liegt, die über zwei Parameter a_0, a_1 beschrieben werden kann (Achsenabschnitt a_0 und Steigung a_1). Dies bedeutet, dass alle Punkte $(x_n, y_n)^t$ auf der Kante der Geradengleichung genügen müssen:

$$y_n = a_0 + a_1 x_n$$

Diese Gleichung lässt sich aber auch als Bedingung für a_0 und a_1 umformulieren:

$$a_1 = \frac{y_n}{x_x} - \frac{1}{x_n} a_0$$

Dies ist wieder eine Geradengleichung, aber mit Achsenabschnitt $\frac{y_n}{x_x}$ und Steigung $-\frac{1}{x_n}$. Außerdem handelt es sich um eine Gerade im sog. Parameterraum, der von a_0 und a_1 aufgespannt wird. Dies kann man so interpretieren, dass jeder Punkt $(a_0, a_1)^t$ im Parameterraum eine bestimmte Gerade im Bild repräsentiert (eben Achsenabschnitt und Steigung). Nun kann man also jedem Punkt auf der Kante eine Gerade im Parameterraum zuordnen. Alle diese Gerade sollten sich (zumindest ungefähr) in einem Punkt treffen. Seine Koordinaten geben dann einen guten Schätzwert für die Geradengleichung der Kante. Das ganze ist in Abb. 15.9 für einige Punkte veranschaulicht.

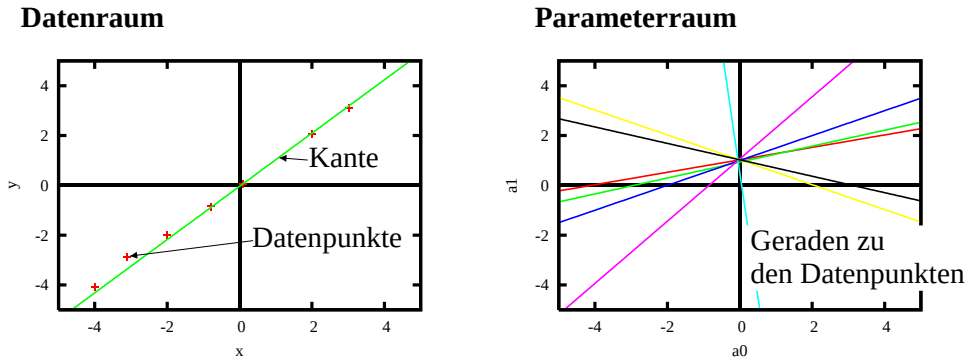


Abb. 15.9: Beispiel zur 1D-Hough-Transformation

In der Praxis verwendet man nicht die Steigung als Parameter (diese kann unendlich werden), sondern den Winkel des Geraden zu den Achsen. Der große Nachteil der Hough-Trafo ist der große Rechenaufwand, da zunächst die Parameter der gerade im Parameterraum berechnet werden müssen und hernach die Schnitte aller geraden miteinander. Aus allen diesen Schnittpunkten muss man schließlich noch den Mittelwert bilden.

schnelle Hough-Transformation: Man kann bei der Berechnung der Hough-Transformation noch zusätzliche Daten berücksichtigen. Wenn ein Punkt auf einer Kante liegt, so kann man über die Methoden aus Kapitel 11 (Strukturtensor etz.) die Orientierung der Kante bestimmen. Man hat also neben der Position der Kante auch ihre Orientierung gegeben. Dies reduziert die Gerade im Parameter-Raum auf einen Punkt, sodass das Auffinden der Schnittpunkte der Geraden im Parameterraum entfällt. Im Parameterraum erhält man dann Cluster von Punkten um die tatsächlichen Parameter.

15.5.3 Kontinuierliche Modellierung: Variationsmethode

Für die Modellierung mit der Variationsmethode geht man von folgender Situation aus: Die Daten seien als $g(\vec{x})$ gegeben. Das Ergebnis soll eine Zielfunktion $f(\vec{x})$ sein, die die Daten möglichst gut beschreibt (z.B. die Umrandung eines Objektes bei der Segmentierung). Man stellt nun ein sog. Fehlerfunktional $\epsilon[f]$ auf. Es misst die Abweichung der Daten von der Zielfunktion. Man wählt dann dasjenige f mit dem kleinsten Fehlerfunktional. Für diskrete Daten und eine euklidische Distanz zwischen f und g führt dies etwa auf einen *Least-Square-Fit*.

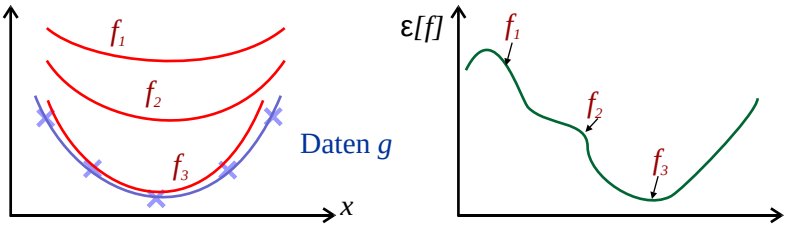


Abb. 15.10: Verschiedene Versuche eine Funktion f an die messdaten zu fitten. rechts ist das Fehlerfunktional gezeigt. Es muss das Minimum aufgesucht werden.

Aufstellen des Fehlerfunktionals bei kontinuierlichen Daten: Wie man an dem Beispiel in Abb. 15.10 sieht, ist es wichtig den Abstand zwischen allen Datenpunkten und f zu messen. Sind die Daten kontinuierlich (durchgezogene blaue Linie), so muss man über beliebig dicht liegende Punkte summieren, man integriert also über den Abstand zwischen g und f. Wenn das Fehlerfunktional nur von f abhängt, so werden keine Nachbarschaftsbeziehungen beachtet. Um diese ebenfalls einzubinden (etwa

um Glattheit zu garantieren) muss das Fehlerfunktional auch von den Ableitungen $\partial_i f = \frac{\partial f}{\partial x_i}$ von f abhängen. Man macht deswegen folgenden allgemeinen Ansatz:

$$\epsilon[f] = \int L(f, \partial_1 f, ..., \partial_W f, x_1, ..., x_W)\, \mathrm{d}^W x \to \min .$$

Dabei ist $\vec{x} = (_1, ..., x_W)$ und W gibt die Dimensionalität der Daten an. Die hier eingeführte Funktion $L(...)$ wird als **Lagrange-Funktion** bezeichnet. Das so dargestellte Optimierungsproblem ist in der Mathematik wohl bekannt. Es wird von denjenigen Funktionen $f(\vec{x})$ gelöst, die den folgenden **Euler-Lagrange-Gleichungen** (oder einfach **Lagrange-Gleichungen**) genügen:

$$\boxed{\frac{\partial L}{\partial f} - \sum_{n=1}^{W} \frac{\partial}{\partial x_n} \left(\frac{\partial L}{\partial(\partial_w f)} \right) = 0}$$

Hierbei handelt es sich um eine partielle Differentialgleichung. Ist f eine vektorwertige Funktion $\vec{f}(\vec{x})$, so gibt es eine Euler-Lagrange-Gleichung für jede Komponente von $\vec{f}(\vec{x})$.

Ähnlichkeitsbedingung: Man fordert, dass die Daten dem Modell möglichst gut entsprechen. Dies kann man etwa über den Abstand nach der L_p-Norm machen. Man erhält dann folgende Lagrange-Funktion:

$$L = \|p\|\, \vec{f}(\vec{x}) - \vec{g}(\vec{x})$$

Daran sieht man, dass die Daten direkt in der Lagrange-Funktion stecken. Nimmt man an, dass das Bild durch die Faltung mit einem Kern h verändert wurde, erhält man:

$$L = \|p\|\, f \circledast h - g$$

Glattheitsbedingung: Oft ist es sinnvoll zu fordern, dass die Funktion $f(x)$ glatt ist. Dies bedeutet, dass $f(f)$ nur langsam mit x variiert. Anschaulich ist dies etwa bei großen Objekten (mehrere Pixel breit) mit glattem Rand gegeben. Wenn eine Funktion glatt ist, so ist ihre Ableitung $\vec{\nabla} f$ klein. Man kann also folgende Bedingung zur Lagrange-Funktion hinzufügen:

$$L = \|p\| \ldots - \alpha^2 \|q\|\, \vec{\nabla} f^2$$

Lösung der Euler-Lagrange-Gleichung: Zur Lösung der so erhaltenen Euler-Lagrange-Gleichungen kann man numerische Verfahren verwenden, die dann die optimale Zielfunktion f ausgeben.

15.5.4 Diffusionsmodelle

Es gibt noch eine andere Herangehensweise an die Euler-Lagrange-Gleichung. Dazu modifiziert man die (im Beispiel 1D-) Lagrange-Gleichung:

$$\frac{\partial L}{\partial f} - \frac{\partial}{\partial x} \frac{\partial L}{\partial(\partial_x f)} = \frac{\partial f}{\partial t}$$

Rechts steht jetzt nicht mehr 0, sondern die zeitliche Ableitung von f. Dies bedeutet, dass die richtige Lösung von f die Gleichgewichtslösung dieses „Reaktions-Diffusions-Systems“ ist. Die Konstante α^2 in der Glattheitsbedingung übernimmt jetzt die Rolle einer Diffusionskonstante. Indem man diese variiert kann man die Diffusion (und damit die Lösung f) an die lokale Umgebung anpassen.

inhomogene Diffusion: Wird etwa α^2 von der Kantenstärke $\left|\vec{\nabla} f\right|^2$ abhängig, so kann man erreichen, dass Kanten weniger geglättet werden, als homogene Bereiche. Die Modellierung würde sich dann gut zum Auffinden von Kanten eignen.

anisotrope Diffusion: Hier wird die Diffusion an die Orientierung von Kanten angepasst, sodass eine Glättung entlang, aber nicht senkrecht zur Kante stattfindet.

16 Morphologie

Aus der Segmentierung eines Bildes erhält man ein Binärbild mit folgender Eigenschaft:

$$g_{mn} = \begin{cases} 1: & \text{Pixel gehört zu einem Objekt} \\ 0: & \text{Pixel gehört nicht zu einem Objekt} \end{cases}$$

Hier sollen nun einige Operationen auf Binärbildern, die morphologischen Operationen, besprochen werden, mit denen Formen in Binärbildern modifiziert und analysiert werden können.

Zur Formulierung der morphologischen Operationen wird die Sprache der Mengen und der Logik verwendet. Die Operationen wirken nur auf Binärbilder, sodass man 1 mit wahr und 0 mit falsch gleichsetzen kann. Zusätzlich wird definiert:

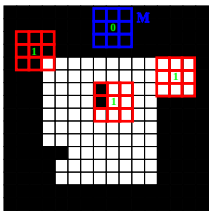
- $\mathbb{M}_q$: Menge der Pixel ungleich null (wahr) in einer Maske um den Punkt q
- $\mathbb{G}$: Menge aller Pixel ungleich null (wahr) im Bild

16.1 Dilatation und Erosion

Dilatation: Diese Operation erweitert die Strukturen in einem bestehenden Binärbild.

Die Abb. 16.1 zeigt dies. Die Objekte werden größer. Die Operation lässt sich so schreiben:

$$\mathbb{G}' = \mathbb{G} \oplus \mathbb{M} = \{p | \mathbb{M}_p \cap \mathbb{G} \neq \varnothing\}$$



Dies bedeutet, dass zum neuen Bild $\mathbb{G}'$ alle diejenigen Pixel p gehören, deren maskierte Umgebung mindestens einen Pixel mit Wert 1 enthält. Dies entspricht der Veroderung aller Pixel der Umgebung:

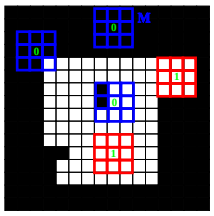
$$g'_{mn} = \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} g_{m+m',n+n'}$$

In dieser Schreibweise entspricht die morphologische Operation einer Faltung, wenn man die Summation durch eine oder-Verknüpfung ersetzt.

Erosion: Diese Operation verkleinert die Strukturen in einem bestehenden Binärbild.

Die Abb. 16.1 zeigt dies. Die Operation lässt sich so schreiben:

$$\mathbb{G}' = \mathbb{G} \ominus \mathbb{M} = \{p | \mathbb{M}_p \subseteq \mathbb{G}\}$$



Dies bedeutet, dass zum neuen Bild $\mathbb{G}'$ alle diejenigen Pixel p gehören, in deren maskierter Umgebung nur Pixel mit Wert 1 enthalten sind. Dies entspricht der Verundung aller Pixel der Umgebung:

$$g'_{mn} = \bigwedge_{m'=-R}^{R} \bigwedge_{n'=-R}^{R} g_{m+m',n+n'}$$

In dieser Schreibweise entspricht die morphologische Operation einer Faltung, wenn man die Summation durch eine und-Verknüpfung ersetzt.

Die Erosion wird zur Entfernung kleiner Elemente des Bildes verwendet (z.B. einzelne Fehlpixel)

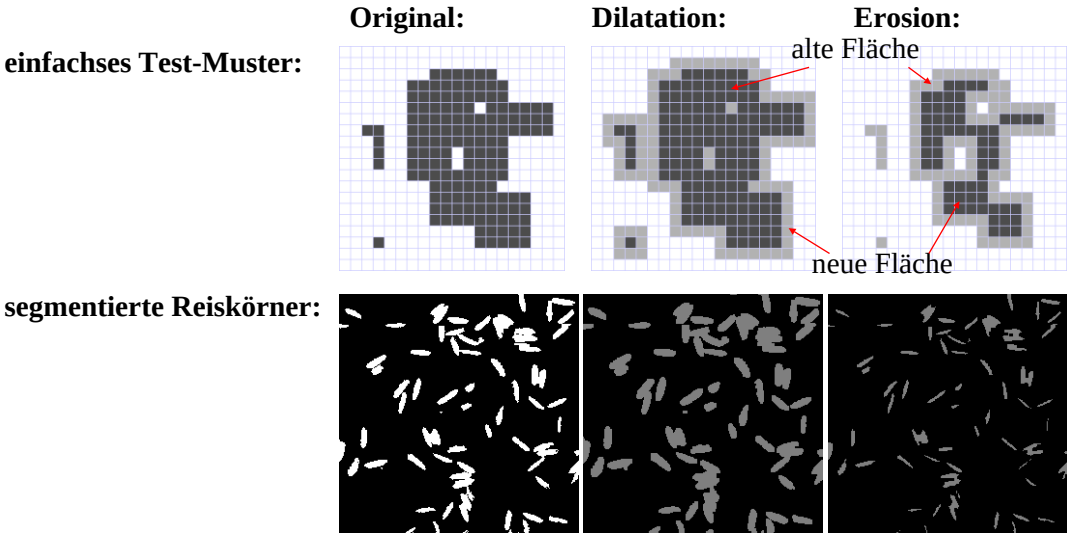


Abb. 16.1: Anwendung von Dilatation und Erosion auf Testbilder

16.2 Allgemeine morphologische Operationen

In Abschnitt 16.1 wurde eine Schreibweise eingeführt, mit der Erosion und Dilatation einer Faltung ähneln (und auch genauso implementiert werden können). Diese Schreibweise soll noch erweitert werden. Dazu definiert man einen Kern h_{mn} der Operation. Alle Pixel dieses Kerns liegen in der lokalen Umgebung $\mathbb{M}$. Man kann dann schreiben:

$$g'_{mn} = \bigvee_{m'=-R}^{R} \bigvee_{n'=-R}^{R} h_{m',n'} \wedge g_{m+m',n+n'}$$

und:

$$g'_{mn} = \bigvee_{m'=-R}^{R} \bigwedge_{n'=-R}^{R} h_{m',n'} \vee g_{m+m',n+n'}$$

Den Kern h_{mn} nennt man auch **Strukturelement**. Damit ergeben sich Dilatation und Erosion als Spezialfälle mit einer Makse, die überall 1 bzw. 0 sind. Wie die LSI-Operatoren auch gehorchen morphologische Operationen $\mathcal{M}$ (Struturelement $\mathbb{M}$) einigen Regeln ($\mathbb{G}$ ist ein Binärbild):

- **Verschiebungsinvarianz:** Mit dem Verschiebungsoperator $\mathcal{S}_{mn}$ gilt:

  $$\mathcal{M}\big(\mathcal{S}_{mn}\mathbb{G}\big) = \mathcal{S}_{mn}\big(\mathcal{M}\mathbb{G}\big)$$

  Die Operation hängt also nicht von der Position im Bild ab.

- **Superpositionsprinzip**:

  $$\mathcal{M}(\mathbb{G} \cup \mathbb{G}') = (\mathcal{M}\mathbb{G}) \cup (\mathcal{M}\mathbb{G}') \quad \text{und} \quad \mathcal{M}(\mathbb{G} \cap \mathbb{G}') = (\mathcal{M}\mathbb{G}) \cap (\mathcal{M}\mathbb{G}')$$

  Für Erosion und Dilatation gelten diese Beziehungen nicht unbedingt.

- **Kommutativität**: Morphologische Operatoren sind i.A. nicht kommutativ:

$$\mathbb{G}_1 \oplus \mathbb{G}_2 = \mathbb{G}_2 \oplus \mathbb{G}_1 \quad \text{aber} \quad \mathbb{G}_1 \ominus \mathbb{G}_2 \neq \mathbb{G}_2 \ominus \mathbb{G}_1$$

  Bei Nacheinander-Anwendung von Erosions- und Dilatationsmaksen auf das selbe Bild gilt Kommutativität:

$$\begin{aligned}(\mathbb{G} \ominus \mathbb{M}_1) \ominus \mathbb{M}_2 &= \mathbb{G} \ominus (\mathbb{M}_1 \oplus \mathbb{M}_2) &= (\mathbb{G} \ominus \mathbb{M}_2) \ominus \mathbb{M}_1 \\ (\mathbb{G} \oplus \mathbb{M}_1) \oplus \mathbb{M}_2 &= \mathbb{G} \oplus (\mathbb{M}_1 \oplus \mathbb{M}_2) &= (\mathbb{G} \oplus \mathbb{M}_2) \oplus \mathbb{M}_1\end{aligned}$$

  Aus diesen Gleichungen kann man etwas über die Implementirung ablesen: werden nacheinander k Strukturelemente $\mathbb{M}_1, ..\mathbb{M}_k$ auf ein Bild angewendet, so ist die äquivalent zur Anwendung des Strukturelement $\mathbb{M}_1 \oplus ... \oplus \mathbb{M}_k$. Dies lässt sich ausnutzen, da man so separierbare Strukturelemente erzeugen kann (ähnlich wie separierbare Filtermasken).

- **Monotonie**: Erosion und Dilation sind monotone Operationen. Dies bedeutet, dass die Teilmengeneigenschaft erhalten bleibt:

$$\begin{aligned}\mathbb{G}_1 \subseteq \mathbb{G}_2 \quad &\Rightarrow \quad \mathbb{G}_1 \oplus \mathbb{M} \subset \mathbb{G}_2 \oplus \mathbb{M} \\ \mathbb{G}_1 \subseteq \mathbb{G}_2 \quad &\Rightarrow \quad \mathbb{G}_1 \ominus \mathbb{M} \subset \mathbb{G}_2 \ominus \mathbb{M}\end{aligned}$$

- **Dualität**: Die Erosion mit dem negierten Bild $\overline{\mathbb{G}}$ ist äquivalent zur Negation der Dilatation mit dem Bild $\mathbb{G}$; und umgekehrt:

$$\overline{\mathbb{G}} \oplus \mathbb{M} = \overline{\mathbb{G} \ominus \mathbb{M}} \quad \text{und} \quad \overline{\mathbb{G}} \ominus \mathbb{M} = \overline{\mathbb{G} \oplus \mathbb{M}}$$

- **Distributivität**:

$$\begin{gathered}(\mathbb{G}_1 \cap \mathbb{G}_2) \oplus \mathbb{M} \subseteq (\mathbb{G}_1 \oplus \mathbb{M}) \cap (\mathbb{G}_2 \oplus \mathbb{M}) \\ (\mathbb{G}_1 \cap \mathbb{G}_2) \ominus \mathbb{M} = (\mathbb{G}_1 \ominus \mathbb{M}) \cap (\mathbb{G}_2 \ominus \mathbb{M}) \\ \text{und} \\ (\mathbb{G}_1 \cup \mathbb{G}_2) \oplus \mathbb{M} = (\mathbb{G}_1 \oplus \mathbb{M}) \cup (\mathbb{G}_2 \oplus \mathbb{M}) \\ (\mathbb{G}_1 \cup \mathbb{G}_2) \ominus \mathbb{M} \supseteq (\mathbb{G}_1 \ominus \mathbb{M}) \cup (\mathbb{G}_2 \ominus \mathbb{M})\end{gathered}$$

16.3 Zusammengesetzte morphologische Operationen

16.3.1 Öffnen und Schließen

Will man mit Erosion Fehlpixel entfernen, so werden dadurch auch die Objekte im Bild kleiner. Man kann deswegen nach der Erosion nochmals eine Dilatation anwenden, die diesen Effekt wieder ausgleicht. Die so entstandene Operation heißt **Öffnen** (**Opening**):

$$\mathbb{G} \circ \mathbb{M} = (\mathbb{G} \ominus \mathbb{M}) \oplus \mathbb{M}$$

Diese Operation kann auch zur Entfernung von Linien verwendet werden, die kleiner als das Strukturelement $\mathbb{M}$ sind.

Mit der Dilatation kann man Löcher in Objekten füllen. Dabei werden aber auch die Objekte größer. Dies kann man durch anschließendes erodieren vermeiden. Man erhält dann eine Operation, die **Schließen** (**Closing**) genannt wird:

$$\mathbb{G} \bullet \mathbb{M} = (\mathbb{G} \oplus \mathbb{M}) \ominus \mathbb{M}$$

Die Operationen Öffnen und Schließen sind „idempotent“, d.h. zwei Anwendung von Schließen direkt hintereinander entsprechen der einfachen Anwendung (genauso für Öffnen):

$$\mathbb{G} \bullet \mathbb{M} = (\mathbb{G} \bullet \mathbb{M}) \bullet \mathbb{M}$$
$$\mathbb{G} \circ \mathbb{M} = (\mathbb{G} \circ \mathbb{M}) \circ \mathbb{M}$$

In Abb. 16.2 sind die Operationen an einem Testbild veranschaulicht. Man sieht deutlich, dass Schließen Löcher in den Objekten schließt, gleichzeitig aber auch das Rauschen etwas verstärkt. Öffnen entfernt das Rauschen. Außerdem werden Linien herausgefilter, die dünner als 3 Pixel (Maskenbreite) sind.

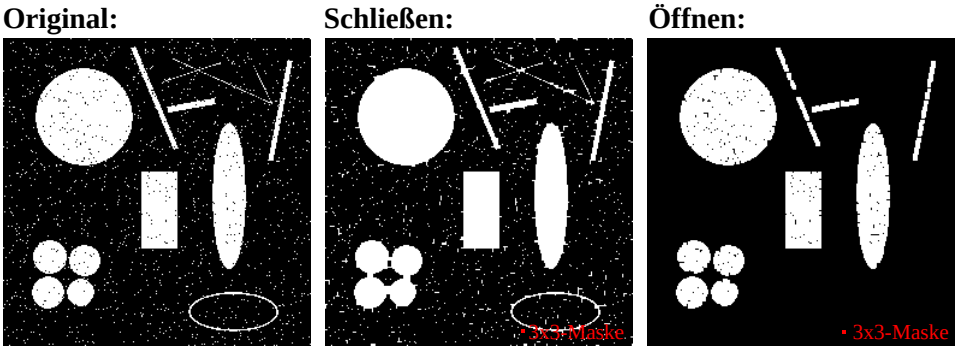


Abb. 16.2: Öffnen und Schließen mit einer 3×3-Maske auf ein Testbild angewendet.

16.3.2 Hit-Miss-Operator

Der Hit-Miss-Operator detektiert bestimmte Strukturen im Bild. Dies soll am Beispiel der Detektion dreier nebeneinander liegender Pixel erläutert werden: Eine Erosion mit dem Strukturelement

$$\mathbb{M}_1 = [1 \;\; 1 \;\; 1]$$

entfernt alle Objekte, die kleiner als drei nebeneinander liegende Pixel sind. In einem zweiten Schritt muss man alle Objekte entfernen, die größer als die drei Pixel sind. Dazu erodiert man den Hintergrund (das invertierte Bild) mit folgender Maske:

$$\mathbb{M}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Dies belässt nur noch diejenigen Objekte im Hintergrund, die kleiner oder gleichgroß zu den drei Pixeln sind. Man kann also aus der Schnittmenge der obigen zwei Teilergebnisse alle Elemente, die ähnlich wie drei Pixel sind herausbekommen. Es bleibt jeweil das zentrale Pixel dieser Objekte stehen. Zusammenfassend erhält man:

$$\mathbb{G} \otimes (\mathbb{M}_1, \mathbb{M}_2) = (\mathbb{G} \ominus \mathbb{M}_1) \cap (\overline{\mathbb{G}} \ominus \mathbb{M}_2) \qquad \text{mit } \mathbb{M}_1 \cap \mathbb{M}_2 = \varnothing$$

Das folgende Maskenpaar detektiert z.B. einzelne Pixel:

$$\mathbb{M}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbb{M}_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

16.4 Extraktion von Rändern

Jeder Randpixel hat mindestens einen Nachbarn, der 0 ist. Die folgenden Erosionsmasken entfernen also alle Pixel, die einen Nachbarn 0 haben, also auf einem Rand liegen (4er und 8er Nachbarschaft):

$$\mathbb{M}_{b4} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \mathbb{M}_{b8} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Das Bild $\mathbb{G}' = \mathbb{G} \ominus \mathbb{M}_b$ entspricht also dem Bild $\mathbb{G}$, allerdings ohne die Randpixel. Entfernt man nun aus dem Bild $\mathbb{G}$ alle Punkte des Bildes $\mathbb{G}'$, so bleiben nur noch die Randpixel übrig:

$$\partial\mathbb{G} = \mathbb{G} \backslash (\mathbb{G} \ominus \mathbb{M}_b)$$

17 Formrepräsentation

Mit den bisherigen Mitteln erhält man immer wieder ein Bild als Ausgabeobjekt. In diesem Kapitel sollen Methode dargestellt werden, mit denen man Objekte über Parameter, wie Größe, Schwerpunkt etz. beschreiben kann.

17.1 Momentbasierte Formmerkmale

Definition 17.1 (Schwerpunkt) *Zur Bestimmung des Schwerpunktes $\overline{x_i}$ eines Grauwertbildes $g(\vec{x})$ in einer Richtung i werden die Positionen der Pixel mit ihrem Grauwert gewichtet:*

$$\overline{x}_i = \frac{\int x_i g(\vec{x})\ \mathrm{d}^2x}{\int g(\vec{x})\ \mathrm{d}^2x}$$

Weiße Pixel tragen also stärker zum Schwerpunkt bei. Besteht ein Objekt aus weißen Pixeln, so kann man hiermit seinen Schwerpunkt berechnen. Die Division durch das zweite Integral führt zur Normierung des Schwerpunktes.

Man kann $g(\vec{x})$ nicht nur als Grauwert auffassen, sondern als beliebiges Merkmal, das umso größer ist, je eher ein Punkt zum Objekt gehört. Im Extremfall kann man das Binärbild aus einer Segmentierung verwenden.

Definition 17.2 (Momente des Grauwertes) *Zur Bestimmung der höheren Momente $\mu_{p,q}$ eines Grauwertbildes $g(\vec{x})$ berechnet man analog zu Statistik:*

$$\mu_{p,q} = \int (x_1 - \overline{x}_1)^p (x_2 - \overline{x}_2)^q\ g(\vec{x})\ \mathrm{d}^2x$$

Für $p = q = 2$ ist dies die Varianz in der Verteilung. Für $p = q = 0$ bleibt noch

$$\mu_{0q} = \int g(\vec{x})\ \mathrm{d}^2x$$

übrig. Dies ist ein Maß für die Fläche des Objektes. Die Momente 1-ter Ordnung $\mu_{0,1}$ und $\mu_{1,0}$ entsprechen den Schwerpunktkoordinaten $\overline{x}_1, \overline{x}_2$.

Mit Hilfe dieser Merkmale kann man Objekte beschreiben. Diese Beschreibung ist aber größenabhängig. D.h. wird das Objekt doppelt so groß, so wird z.B. $\mu_{2,2}$ vervierfacht. Will man unterschiedlich große Objekte vergleichen, so kann man die Momente der Objekte mit dem 0-ten Moment μ_0 normieren:

$$\overline{\mu}_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{(p+q+2)/2}}$$

Zusammenfassend kann man sagen:

- Moment 0. Ordnung: Fläche des Objektes
- Moment 1. Ordnung: Position des Objektes (Mittelwert/Schwerpunkt)

- Moment 2. Ordnung: Ausdehnung des Objektes (Varianz)

Die Momente 2-ter Ordnung $\mu_{1,1}, \mu_{2,0}, \mu_{0,2}$ beschreiben die Ausdehnung des Objektes. Sie entsprechen der Varianz in der Statistik. Sie lassen sich zu einem symmetrischen Tensor (Matrix), dem Trägheitstensor $\mathbf{J}$ zusammenfassen:

$$\mathbf{J} = \begin{pmatrix} \mu_{2,0} & -\mu_{1,1} \\ -\mu_{1,1} & \mu_{0,2} \end{pmatrix}$$

Dieser Tensor hat die Eigenschaft, dass das Koordinatensystem, in dem er Diagonalgestalt hat (Basisvektoren sind Eigenvektoren von $\mathbf{J}$) in den Hauptachsen des Objektes liegt. Die folgende Abb. 17.1 verdeutlicht dies. Die Momente zweiter Ordnung geben die Ausdehnung der eingezeichneten Ellipse an, dem sog. Trägheitsellipsoiden. Sie modellieren das Objekt also im wesentlichen als Ellipse. Das grüne Koordinatensystem entsteht, wenn der Tensor diagonalisiert wird. Seine Verkippung θ zum schwarzen System (beide liegen im Schwerpunkt) gibt die Neigung des Objektes an.

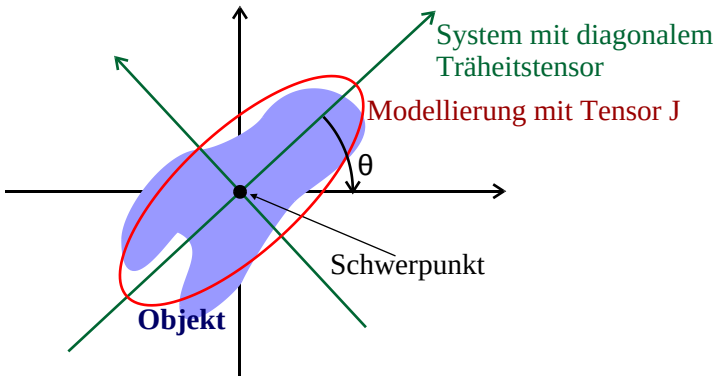


Abb. 17.1: Trägheitstensor eines Objektes

Man kann den Winkel θ und die Exzentrizität ϵ berechnen:

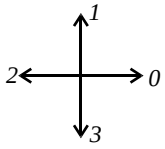
$$\theta = \frac{1}{2} \arctan \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}, \qquad \epsilon = \frac{(\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2}{(\mu_{2,0} + \mu_{0,2})^2}$$

Die Exzentrizität gibt an, wie stark das Objekt von einem Kreis ($\epsilon = 1$) abweicht. Sie ist ein Maß für das Verhältnis der Hauptachsen des Trägheitsellipsoiden.

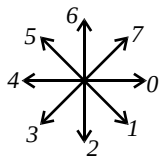
17.2 Richtungsketten

Bisher wurde ein Bild über seine Pixelmatrix dargestellt. Wenn man aber in einem Schwarz-Weiß-Bild einzelne, abgegrenzte Objekte hat, so genügt es auch einfach deren Umriss abzuspeichern. Gerade wenn die Objekte groß sind spart man so sehr viel Speicher ein. Dazu wird von einem Startpunkt aus (gegen den Uhrzeigersinn) immer der nächste Punkt des Randes gesucht. Es gibt dabei zwei mögliche Nachbarschaftsbeziehungen:

4er-Nachbarschaft:



8er-Nachbarschaft:



Mit einer 4er-Nachbarschaft benötigt man also 2 Bit und mit einer 8er-Nachbarschaft 3 Bit um den nächsten Pixel zu codieren. Die folgende Abb. 17.2 zeigt ein Beispiel. Der Kettencode hat auch den Vorteil, dass eine Fläche L^2 Pixel umfasst, als ein 2D-Objekt ist, während ihre Umrandung nur $\propto R$ Pixel enthält. Man spart also eine Dimension bei der Kodierung.

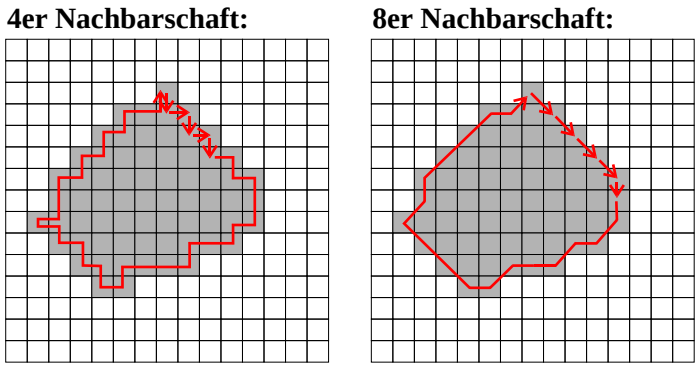


Abb. 17.2: Kettencodierung eines Objektes mit verschiedenen Nachbarschaften

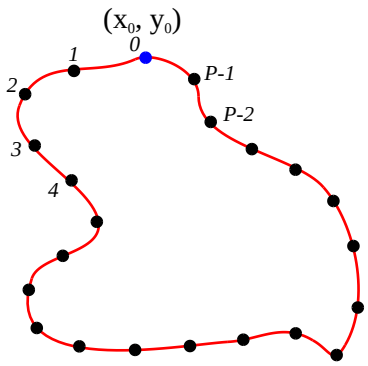


Abb. 17.3: äquidistante Abtastung einer 2D-Kurve (rot) mit $P = 21$ Punkten

17.3 Fourierdeskriptoren

In Abschnitt 17.2 über Richtungsketten wurde beschrieben, wie man eine Kurve (die Umrandung einer Fläche/eines Objektes) geschickt kodieren kann. Hier soll nun ein Ansatz beschrieben werden, wie man eine solche Kurve beschreiben kann. Als Ausgangsdaten verwendet man eine Kurve, die die Umrandung des Objektes beschreibt. Diese Kurve wird an diskreten Punkten abgetastet, sodass die Pfadlänge (!) zwischen zwei Punkten immer p beträgt. In Abb. 17.3 ist dies an einem Beispiel gezeigt. Jeder Punkt in dieser Serie von Abtastungen wird durch zwei Zahlen (Koordinaten) beschrieben: $x(p), y(p)$. Diese zwei reellen Koordinaten können zu einer komplexen Zahl zusammengefasst werden:

$$z(p) = x(p) + \mathrm{i} \cdot y(p)$$

Die Kurve ist zyklisch, d.h.

$$z(p + nP) = z(p), \qquad n \in \mathbb{Z}$$

Dies entspricht der periodischen Fortsetzung der Zahlenreihe $z(p)$. Man hat damit eine periodische Folge von komplexen Zahlen. Diese kann auch als periodisches Zeilenbild aufgefasst werden und man kann die Fourier-Transformierte dieser Folge berechnen:

$$\hat{z}_\nu = \frac{1}{P} \int_0^P z(p) \cdot \exp\left(\frac{-2\pi \mathrm{i} \nu p}{P}\right) \, \mathrm{d}p$$

Die Fourierkoeffizienten $\hat{z}_\nu$ werden als **Fourierdeskriptoren** bezeichnet. Aus ihnen kann man die Kurve rekonstruieren:

$$z(p) = \sum_{\nu=-\infty}^{\infty} \hat{z}_\nu \exp\left(\frac{2\pi \mathrm{i} \nu p}{P}\right) \qquad \rightarrow \quad x(p)\mathrm{Re}\,[z(p)], \;\; y(p)\mathrm{Im}\,[z(p)]$$

geometrische Deutung: Der erste Fourierdeskriptor lautet:

$$\hat{z}_0 = \frac{1}{P}\int_0^P z(p)\,\mathrm{d}p = \frac{1}{P}\int_0^P x(p)\,\mathrm{d}p + \frac{\mathrm{i}}{P}\int_0^P y(p)\,\mathrm{d}p = \overline{x} + \mathrm{i}\overline{y}$$

Dies ist gerade der Schwerpunkt, bzw. Mittelpunkt der Kurve. Hat nur der zweite Desktiptor einen Beitrag, so wird aus obiger Summe:

$$z(p) = \hat{z}_1 \exp\left(\frac{2\pi \mathrm{i}p}{P}\right) = \underbrace{r_1 \cdot \mathrm{e}^{\mathrm{i}\varphi_1}}_{=\hat{z}_1 \in \mathbb{C}} \cdot \mathrm{e}^{2\pi \mathrm{i}p/P} = r_1 \cdot \mathrm{e}^{\mathrm{i}\varphi_1 + 2\pi \mathrm{i}p/P}$$

Dies beschreibt gerade einen Kreis mit Radius r_1. Der Deskriptor $\hat{z}_{-1}$ beschreibt ebenfalls einen Kreis, der aber in der entgegengesetzten Richtung umläuft. Zusammen beschreiben sie eine Ellipse:

$$\hat{z}_1 + \hat{z}_{-1} = (r_1 + r_{-1}) \cdot \cos(2\pi p/P) + \mathrm{i} \cdot (r_1 - r_{-1}) \cdot \sin(2\pi p/P)$$

So fügen die Deskriptoren höherer Ordnung immer mehr Details zur Kurve hinzu.

polare Fourierdeskriptoren: Alternativ zur Abtastung mit äquidistanten Punkten kann man auch den Abstand des Randes zum Mittelpunkt in äquidistanten Winkelabständen messen. Dies führt auf eine Serie von Abständen r_n, aus denen man dann die sog. polaren Fourierdeskriptoren berechnen kann.

Objektsymmetrien: Hat das Objekt eine m-zahlige Rotationssymmetrie (z.B. Quadrat: $m = 4$, Linie: $m = 2$, gleichseitiges Dreieck: $m = 3$), so tragen nur die Fourierdeskriptoren $z_{1\pm vm}$ bei. Die andere sind 0. Für eine Linie tragen z.B. nur $z_{\pm 1}, z_{\pm 3}, z_{\pm 5}, \ldots$ bei. Bei einem Quader nur $z_{-7}, z_{-3}, z_1, z_{\pm 5}, z_{\pm 9}, \ldots$.

Invariante Objektbeschreibung:

- **positionsunabhängige Beschreibung:** Die Position eines Objektes beeinflusst nur den Koeffizienten $\hat{z}_0$.
- **größenunabhängige Beschreibung:** Wird ein Objekt um den Faktor α vergrößert, so werden auch alle Fourierdeskriptoren $\nu \geq 1$ mit α multipliziert. Man kann also alle Deskriptoren auf den Deskriptor $\hat{z}_1$ normieren. Danach sind sie alle größenunabhängig und es gilt $\hat{z}_1 = 1$.
- **ausrichtungsunabhängige Beschreibung:** Wird das Objekt um den Winkel φ gedreht, so erhält jeder Deskriptor $\hat{z}_\nu$ eine zusätzliche Phase $\hat{z}_\nu \rightarrow \hat{z}_\nu \cdot \mathrm{e}^{\mathrm{i}\varphi}$. Man kann diese Phase also aus allen Deskriptoren herausrechnen (normieren auf die Phase von $\hat{z}_1$) und erhält so eine rotationsunabhängige Beschreibung.

Sämtliche wichtigen Invarianzen liegen also in den ersten beiden Deskriptoren $\hat{z}_0$ und $\hat{z}_1$.

Nachteile: Der einzige Nachteil der sehr eleganten Fourierdeskriptoren ist, dass sie gut bestimmte und äquidistante (!) Punkte (x_i, y_i) auf dem Rand benötigen. Die Punkte, die sich aus der 4er- und 8er-Nachbarschaft der Kettenkodierung (siehe Abschnitt 17.2) ergeben sind leider für 8er-Nachbarschaften nicht äquidistant, sodass sie nicht als Ausgangsdaten taugen. Die Verwendung der 4er-Nachbarschaft ist zwar möglich, der Rand wird dadurch aber sehr ausgefranst. Deswegen müssen die Punkte auf Subpixel-Genauigkeit bestimmt werden, was leider nur schwer möglich ist.

17.4 Formparameter

Oft möchte man einfache Parameter eines Objektes bestimmen, wie z.B. Fläche, Umfang oder Rundheit. Dies ist mit den obigen Codierungen sehr einfach. Die Bestimmung dieser Parameter aus den verschiedenen Bildrepräsentationen soll hier beschrieben werden:

Fläche: In einem Pixelbild muss man zur Flächenberechnung nur die Bildpunkte im Objekt zählen. Ist ein Objekt kettenkodiert kann man einfach numerisch über die Fläche integrieren. Man legt dazu eine Referenzlinie fest und läuft die einzelnen Bildpunkte ab. Man berechnet dann die Entfernung B_i zu dieser Referenzlinie (in Pixeln) und addiert sie zur Fläche. Dann geht man zum nächsten Pixel über. Geht man dabei nach rechts unten, so gilt $B_{i+1} = B_i - 1$ geht man nach rechts, so ist $B_{i+1} = B_i$ usw. Geht man nach unten, so ist $B_{i+1} = B_i - 1$, die Fläche nimmt aber nicht zu. In Abb. 17.4 ist das veranschaulicht.

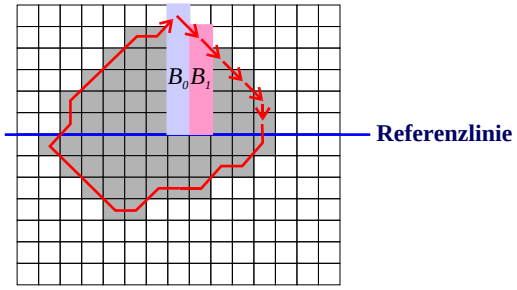


Abb. 17.4: Flächenberechnung mit Kettencode

In der Beschreibung mit Fourierdeskriptoren ergibt sich die Fläche zu:

$$A = \pi \sum_{\mu=-N/2}^{N^2-1} \nu \cdot |\hat{z}_\nu|^2$$

Umfang: Der Umfang ist aus der Kettenkodierung am einfachsten zu berechnen, indem man die Kette einfach entlang läuft und die Abstände aufaddiert. Aus den Fourierdeskriptoren lässt sich der Umfang nicht direkt berechnen. Es ist zu beachten, dass der Umfang vom Rauschen im Bild abhängt, weil der Rand bei der Segmentierung umso ausgefranster (länger) wird, je größer das Bildrauschen ausfällt.

Rundheit: Die Rundheit c eines Objektes ist ein größenunabhängiger Formparameter. Er ergibt sich aus Umfang U und Fläche A:

$$c = \frac{U^2}{A}$$

Die Rundheit setzt den Umfang mit der Fläche in Beziehung. Für ein ideal rundes Objekt (einen Kreis) erhält man $c_k = \frac{4\pi^2 r^2}{\pi r^2} = 4\pi \approx 12.6$ Für ein Quadrat nur noch $c_q = \frac{(4L)^2}{L^2} = 16$. Je runder ein Objekt ist, desto kleiner wird c. Der Minimalwert ist derjenige des Kreises.

18 Klassifikation

18.1 Problemstellung

Bisher wurden Bilder soweit verarbeitet, dass sich Merkmale ergeben anhand derer Regionen und Objekte im Bild in verschiedene Klassen eingeordnet werden können. Ein Beispiel ist etwa verschiedene Objekte in einem Bild zu unterscheiden. Zunächst wird ein solches Bild vorverarbeitet. Dabei werden die Objekte O_i im Bild segmentiert. Danach werden beschreibende Parameter der Objekte bestimmt, wie etwa die Größe und andere Formparameter oder die Parameter der Textur auf den Objekten. Diese Eigenschaften der Objekte werden zu sog. **Merkmalsvektoren** $\vec{m}_i$ zusammengefasst. Die einzelnen Merkmale spannen damit den **Merkmalsraum** auf. Jedem Objekt o_i ordnet man also einen Punkt (Vektor) $\vec{m}_i$ dieses Raumes zu. Die Aufgabenstellung ist nun die verschiedenen Objektklassen im Merkmalsraum zu unterscheiden. Wenn die Merkmale gut gewählt sind, so sollten sich ähnliche Objekte nahe beieinander im Merkmalsraum einfinden (**Kluster**). Das ist in Abb. 18.1 schematisch für ein und zwei Merkmale illustriert.

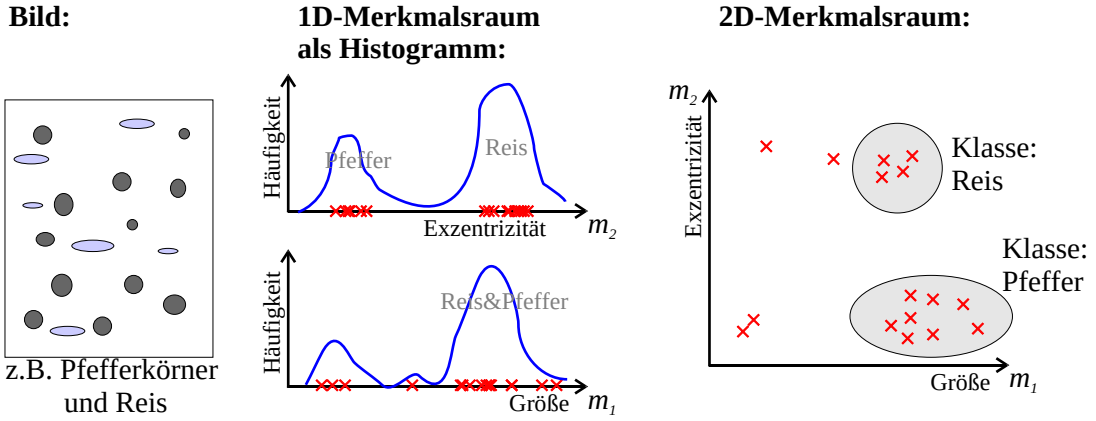


Abb. 18.1: Merkmalsraum für ein einfaches Klassifikationsbeispiel von einem Bild mit Reis- und Pfefferkörnern

Es zeigt sich im 2D-Merkmalsraum (aufgespannt von den Formparameter Größe/Fläche und Exzentrizität), dass die zwei Klassen Reis und Pfeffer gut zu unterscheiden sind (bis auf einige Ausreißer. Die 1D-Merkmalsräume zeigen beide eine bimodale Verteilung. Es zeigt sich aber, dass nur die Exzentrizität zur Klassifikation herangezogen werden kann, weil Reis und Pfeffer in etwa die selbe Größe haben.

An diesem Beispiel sieht man die notwendigen Aufgaben bei der Klassifizierung in der Bildverarbeitung:

1. Bildaufnahme
2. Bildvorverarbeitung
3. Identifizieren von sinnvollen Merkmalen (Der Versuch rechteckige und runde Papierschnipsel anhand der Farbe zu unterscheiden wird scheitern)
4. Auswahl eines Klassifikators
5. evtl. Training des Klassifikators mit Trainigsdatensatz und Evaluation des Trainings
6. Einsatz des Klassifikators auf die zu verarbeitenden Daten.

Selektion von Merkmalen: Es gilt gute Merkmale zur Unterscheidung zu finden. Im obigen Beispiel würde wohl auch die Exzentrizität zur Unterscheidung ausreichen, da die Größe keine sinnvolle neue Information liefert. Allgemein ist es nicht so, dass die Güte der Klassifikation mit mehr Parametern automatisch zunimmt. Oft wird man bestimmte Klassen trotz einer idealen Merkmalswahl nicht unterscheiden können. Dies kann z.B. auch daran liegen, dass es keine eindeutige Klassifizierung gibt. So können Zellen zwar normal oder pathologisch sein, es können aber auch beliebige Abstufungen auftreten. U.U. deutet eine unmögliche Klassifizierung auch auf unzureichende Bildaufnahme hin. Es ist also immer nötig auch zu untersuchen, ob eine Klassifizierung überhaupt nötig ist. Es ist auch zu beachten, dass die Komplexität und Laufzeit des Klassifikators mit der Anzahl der Merkmale ansteigt.

Hauptachsentransformation: Man betrachte folgende Skizze in Abb. 18.2. Sie zeigt einen Merkmalsraum aus zwei Merkmalen m_1, m_2 und eine Menge von Punkten, die klassifiziert werden sollen.

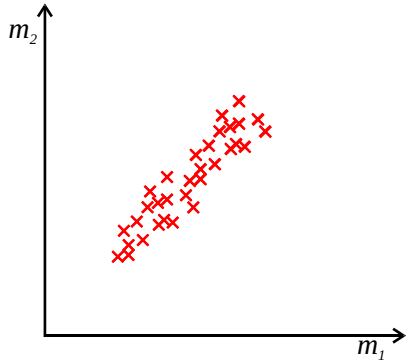


Abb. 18.2: Merkmalsraum mit korellierten Merkmalen

In diesem Raum sind die Merkmale offensichtlich korreliert. Ein hoher m_1-Wert impliziert auch einen hohen m_2-Wert. Um dies mathematisch auszudrücken betrachtet man die **Kreuzkovarianzen** σ_{pq} der Stichprobe, sowie die **Varianzen** σ_{pp}:

$$\sigma_{pp} = \overline{\left(m_p - \overline{m}_p\right)^2} \qquad \sigma_{pq} = \overline{\left(m_p - \overline{m}_p\right) \cdot \left(m_q - \overline{m}_q\right)}$$

Die Kovarianz σ_{pq} ist ein Maß für die Korrelation der Merkmale m_p und m_q. Die Varianz σ_{pp} ist ein Maß für die Streuung der Merkmale. Ein gutes Merkmal sollte eine kleine Varianz aufweisen, weil dies auch kleine Kluster impliziert. Die σ_{pq} lassen sich zur sog. **Kovarianzmatrix** zusammenfassen:

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1P} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{P1} & \sigma_{P2} & \cdots & \sigma_{PP} \end{pmatrix}$$

Bei unkorrelierten Merkmalen hat diese Matrix Diagonalgestalt. Bei korrelierten Merkmalen tragen auch die Kovarianzen bei. Man kann nun das Koordinatensystem im Merkmalsraum so drehen, dass Σ Diagonalgestalt hat. Dies ist in Abb. 18.3 veranschaulicht.

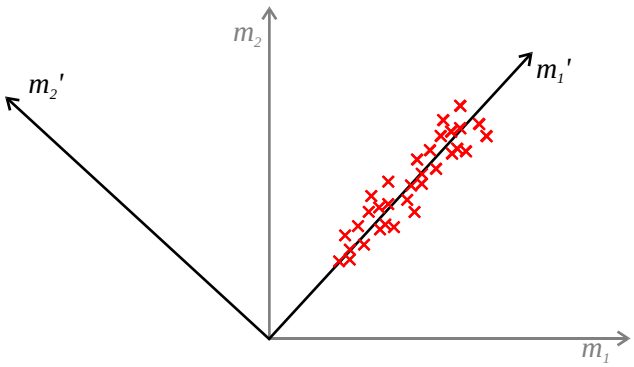


Abb. 18.3: Merkmalsraum mit korrelierten Merkmalen nach der Hauptachsentransformation

Es zeigt sich nun, dass das neue Merkmal m_1' keine signifikante Aussage zur Klassifikation beitragen kann. Man kann so den Merkmalsraum auf das neue Merkmal m_2' einschränken. Seine Varianz ist relativ gering und die Verteilung damit eng und scharf. Es handelt sich also um ein gutes Merkmal.

überwachte Klassifizierung: Wird der Klassifikator mit einem Trainingsdatensatz vor der eigentlichen Klassifizierung trainiert, so spricht man von einem überwachten Verfahren. Dazu verwendet man einen Datensatz, bei dem man die Klassifizierung bereits kennt und optimiert damit die Trennung der Klassen im Merkmalsraum.

unüberwachte Klassifizierung: Hierbei berechnet man direkt die Merkmalsvektoren der zu klassifizierenden Objekte und analysiert danach die Punkteverteilung im Merkmalsraum. Dieser Ansatz ist sicher objektiver (weil ohne Anwendung von Vorwissen), dafür ist die Trennung möglicherweise weniger gut. Hierbei muss auch die Anzahl der Klassen vorher nicht bekannt sein.

selbstlernende Klassifizierung: Hierbei wird die Trennung im Merkmalsraum mit jedem neuen, zu klassifizierenden Objekt angepasst. Ein solcher Ansatz hat den Vorteil, dass er nicht steif vorgegeben ist, sich also an dynamische Situationen (zeitabhängige Daten, Beleuchtung etz.) anpassen kann.

18.2 Einfache Klassifikationsverfahren

Definition 18.1 (Klassifikator) *Ein Klassifikator k ist eine Funktion, die vom Merkmalsraum $\mathbb{M}$ in den Entscheidungsraum $\mathbb{D}$ abbildet.*

$$k: \mathbb{M} \mapsto \mathbb{D}$$

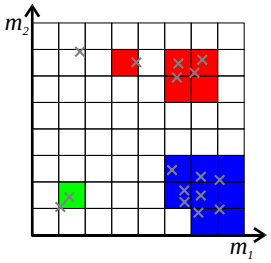
Der Entscheidungsraum ist ein Raum mit Q Elementen, zu jeder möglichen Klasse gehört ein Element. Im einfachsten Fall kann jedes Element die Werte 0 und 1 annehmen. Ordnet nun k ein Objekt mit dem Merkmalsvektor $\vec{m} \in \mathbb{M}$ einer Klasse q zu, so ist das entsprechende Element in $\mathbb{D}$ 1 und alle anderen 0. Eine weitere Möglichkeit wäre ein Wertebereich $[0..1]$, wobei jeweils die Wahrscheinlichkeit für die Zugehörigkeit zu einer Klasse angegeben wird.

Jedes der folgenden Verfahren kann für überwachtes und unüberwachtes Klassifizieren eingesetzt werden. Diese zwei Ansätze unterscheiden sich ja nur dadurch, wie die Einteilung des Merkmalsraumes in Klassen vonstatten geht, nicht darin, wie die Klassifizierung eines neuen Objekte $\vec{m} \in \mathbb{M}$ funktioniert.

18.2.1 Nachschaumethode

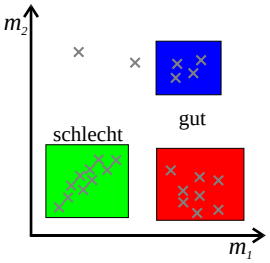
Dies ist das einfachste Klassifizierungsverfahren. Der Merkmalsraum wird in diskrete Parzellen eingeteilt. Jeder Parzelle des Merkmalsraumes wird eine Klasse zugeordnet. Überschneiden sich zwei Klassen an einem Punkt wählt man z.B. diejenige mit der höchsten Zugehörigkeitswahrscheinlichkeit aus. Ein neues Objekt mir Vektor $\vec{m} \in \mathbb{M}$ wird dann je nach der getroffenen Parzelle klassifiziert. Die meisten Parzellen werden wohl aber keiner Klasse zugeordnet werden.

Für niedrig-dimensionale Räume $\mathbb{M}$ ist dies ein sehr schnelles Verfahren, da nur ein Tabellen-Lookup nötig ist. Dafür ist aber evtl. der Speicherverbrauch für die Lookup-Tabelle sehr groß. Angenommen es gibt 3 Merkmale und 64 Parzelle Pro Achse. Wird zu jeder Parzelle ein Byte gespeichert, so benötigt man bereits $64^3 = 0.256\ \mathrm{MByte}$ für die Tabelle.



18.2.2 Quadermethode

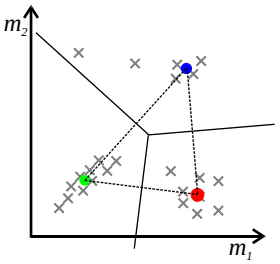
Hierbei wird jede Klasse im Merkmalsraum von einem Quader umgeben. Ist ein neuer Vektor $\vec{m} \in \mathbb{M}$ innerhalb eines Quaders, so wird er als zugehörig klassifiziert. Liegt er außerhalb aller Quader, so gehört er zu keiner Klasse. Die Umschreibung von Quadern um Punktverteilungen funktioniert für runde Verteilungen gut. Sind die Merkmale aber korreliert, so ist diese Methode eher schlecht (→ Hauptachsentransformation). Pro Dimension und Klasse werden hier 2 vergleiche benötigt. Also braucht man bei N Dimensionen und Q Klassen etwa $2 \cdot P \cdot Q$ Vergleiche.



18.2.3 Methode des geringsten Abstandes

Bei dieser Methode wird jede Klasse durch ihren Schwerpunkt $\vec{m}_q$ repräsentiert. Ein Objekt $\vec{m} \in \mathbb{M}$ gehört dann zu der Klasse, zu der es den geringsten (euklidischen) Abstand hat:

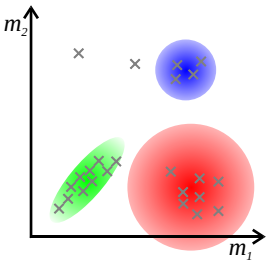
$$Q = \min_q |\vec{m} - \vec{m}_q|^2$$



Dieses Verfahren führt zur rechts dargestellten Einteilung des Raumes. Die Begrenzungslinien der Cluster stehen senkrecht auf den Verbindungslinien der Schwerpunkte. Dieses Verfahren kann leicht abgewandelt und optimiert werden. So kann man etwa einen Skalierungsfaktor in die Abstandsberechnung einführen, der dazu führt, dass der notwendige Abstand zu wenig ausgedehnten Klustern geringer ist, als zu großen Klustern. Oder man definiert einen Maximalabstand, der $\vec{m}$ zu einem Zentrum haben darf. Liegt er außerhalb aller Maximalabstände, so gehört er zu keiner Klasse.

18.2.4 Methode der höchsten Wahrscheinlichkeit

Diese Methode modelliert jede Klasse als Wahrscheinlichkeitsverteilung (z.B. multivariate Normal-Verteilungen). Zu einem neuen Objekt $\vec{m} \in \mathbb{M}$ werden dann die Wahrscheinlichkeiten p_i berechnet, dass es zur Klasse i gehört. Es wird dann entweder in die Klasse mit der höchsten Wahrscheinlichkeit eingeordnet, oder es werden einfach die Wahrscheinlichkeiten zurückgegeben, um sie weiter auszuwerten.



Teil V

Mathematische Formelsammlung

19 Komplexe Zahlen

Zunächst sind die Bilder, die eine Kamera liefert ja reelle Bilder, d.h. Jedem Bildpunkt $\vec{x}$ wird eine reelle Zahl $g(\vec{x}) \in \mathbb{R}$ zugeordnet. Oft ist es aber nötig auch komplexe Werte darzustellen. Hier sind deswegen kurz einige Rechenregeln für komplexe Zahlen zusammengestellt:

1. **Komplexe Einheit i:**

$$i := \sqrt{-1} \qquad \Rightarrow i^2 = -1, \quad \frac{1}{i} = -i \tag{19.0.1}$$

2. **Normalform einer Zahl $z \in \mathbb{C}$:** Jede komplexe Zahl lässt sich als Summe darstellen:

$$z = a + ib, \qquad a, b \in \mathbb{R}, \quad z \in \mathbb{C} \tag{19.0.2}$$

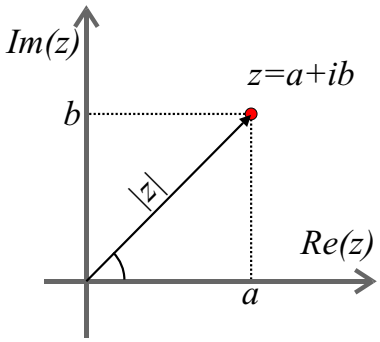


Abb. 19.1: Komplexe Zahlenebene

Dies entspricht einem Punkt (a, b) in der komplexe Zahlenebene (siehe Abb. 19.1). Es werden noch die folgenden Operatoren eingeführt:

$$\mathrm{Re}\,\{a + ib\} = a, \qquad \mathrm{Im}\,\{a + ib\} = b \tag{19.0.3}$$

3. **Polarform einer Zahl $z \in \mathbb{C}$:**

$$z = a + ib = |z| \cdot (\cos\varphi + i \cdot \sin\varphi) = |z| \cdot e^{i\varphi}, \quad absz := \sqrt{a^2 + b^2} \tag{19.0.4}$$

Die Interpetation des Winkels φ liefert Abb. 19.1. er berechnet sich zu:

$$\tan\varphi = \frac{y}{x} \tag{19.0.5}$$

4. **Addition komplexer Zahlen:**

$$z_1 + z_2 = a_1 + a_2 + i \cdot (b_1 + b_2) \tag{19.0.6}$$

5. **Multiplikation komplexer Zahlen:**

$$z_1 \cdot z_2 = a_1 a_2 - b_1 b_2 + i \cdot (a_1 b_2 + a_2 b_2) = |z_1| \cdot |z_2| \cdot e^{i(\varphi_1 + \varphi_2)} \tag{19.0.7}$$

6. **Komplexe Konjugation:**

$$z^* = (a + ib)^* := a - ib \tag{19.0.8}$$

Damit gelten die folgenden Rechenregeln:

$$z^* \cdot z = |z|^2 \tag{19.0.9}$$

$$z + z^* = 2 \cdot \mathrm{Re}\,\{z\}, \qquad z - z^* = 2i \cdot \mathrm{Im}\,\{z\} \tag{19.0.10}$$

7. **Betrag von Summen:** In der Quantenmechanik treten manchmal Ausdrücke der Form $|z_1 + z_2|^2$ auf. Dafür gilt:

$$|z_1 + z_2|^2 = |a_1 + ib_2 + a_2 + ib_2|^2 = a^2+c^2+2ac+b^2+d^2+2bd = a^2+b^2+c^2+d^2+2(ac+bd) = \\ = |z_1|^2 + |z_2|^2 + \mathrm{Re}\,\{z_1 \cdot z_2^*\} \tag{19.0.11}$$

Etwas allgemeiner erhält man:

$$\left|\sum_i z_i\right|^2 = \sum_i |z_i|^2 + \sum_i \sum_{j \neq i} \mathrm{Re}\,\{z_i \cdot z_j^*\} \tag{19.0.12}$$

Anhang

Abbildungsverzeichnis

1.1 Übersicht über die einzelnen Schritte einer Klassifizierung in der Bildverarbeitung . . . 8

2.1 Beispiele für RGB-Farben . . . 10
2.2 verschiedene raumfüllende 2D-Gitter . . . 11
2.3 2D-Nachbarschaftsbeziehungen . . . 12

3.1 1D-Sinus- und Cosinus-Schwingungen, mit unterschiedlicher Phase und Frequenz. . . . 13
3.2 zur Definition des 2D-Wellenvektors . . . 14
3.3 Wellenstrukturen in 2D-Bildern, mit verschiedenen Frequenzen und Ausrichtungen . . . 16
3.4 Fourier-Reihendarstellung einer Sägezahnfunktion (grün: Sägezahn, rot: Fourierreihe). Es werden die ersten ein bis vier Koeffizienten berücksichtigt. . . . 18
3.5 alle Basisfunktionen in Real- und Imaginärteil für eine $N = 16$ DFT . . . 20
3.6 Mit der DFT können nur periodische Strukturen bis zu einer gewissen Grenzfrequenz (2 Bildpunkte pro Schwingung) dargestellt werden. . . . 20
3.7 Real- und Imaginärteil der Basismatritzen für eine 16×16-2D-DFT . . . 22
3.8 Auswirkung der vernachlässigung von Real- und Imaginäreteil, bzw. Betrag oder Phase im Fourierraum. . . . 23
3.9 glatte und weniger glatte Funktion (oben) und ihre kompakte bzw. ausgedehnte Fourier-Transformierte . . . 25
3.10 geometrische Interpretation der Periodizität des Kerns der 1D-DFT . . . 26

4.1 Abtastung mit einer CCD-Kamera . . . 29
4.2 Fehler bei der 1D-Abtastung (Aliasing-Effekte) . . . 30
4.3 Falte eines Vorhangs (dünnes Gitter des Stoffes), mit Moiré-Mustern (mit freundlicher Genehmigung von Julia Ziegler) und Moiré-Effekte beim Rasterdruck . . . 31
4.4 Quantisierung eines Bildes mit unterschiedlich vielen Grau- und Farbstufen . . . 33

5.1 Quadtree eines Bildes . . . 35

6.1 Spreizung des Grauwertbereiches eines Bildes . . . 38
6.2 Spreizung des Grauwertbereiches mit einer γ-Transformation . . . 39
6.3 Effekt der Mittelung von N verrauschten Bildern . . . 40
6.4 Kalibrierung des Absorbtionsbildes eines Bose-Einstein-Kondensates (dunkler Fleck) mit einem Referenzbild, aber ohne Dunkelbild ($d_{mn} = 0$). Die Ringe im Referenz- und Originalbild sind die die Störung durch das Abbildungssystem, die herausgefiltert werden sollen. . . . 40
6.5 Effekt eines Sinus-Fensters auf die FFT . . . 41
6.6 verschiedene geometrische Transformationen . . . 41

7.1 Beispiel einer Faltungsmaske (Laplace-Filter) und anschauliche Darstellung der Faltungsoperation . . . 43
7.2 Prinzip verschiedener Randbedingungen und die Fehler, die dabei auftreten können. Es wurde ein 3×3 Mittelwert-Filter eingesetzt . . . 44
7.3 Medianfilter: Prinzip und Anwendung auf ein Testbild . . . 47
7.4 Minimums- und Maximums-Filter . . . 48

8.1 Darstellung von 1D-Daten und Bilddaten in verschiedenen Auflösungen 50
8.2 Veranschaulichung der Berechnung der Laplace-Pyramide 52
8.3 Erzeugung eines Skalenraumes durch Diffusion am Beispiel eines zufälligen und eines perioschen 1D-Signals 54

9.1 Anwendung von 1D- und 2D-Rechteckfiltern. Im 2D-beispiel wurde der Filter rechts oben und links unten angewendet. Der Rest des Bildes zeigt das Ausgangsmuster. 57
9.2 Transferfunktionen von 1D- und 2D-Rechteckfilter verschiedener Breite 58
9.3 Binomialverteilung für $p = \frac{1}{2}$ 59
9.4 Transferfunktion verschiedener Binomialfilter 60

10.1 differentielle Merkmalsbeschreibung 64
10.2 Kombination von Glättung und Ableitung in einem Filter 66
10.3 Darstellung der Punkte mit Betrag 1 zur euklidischen und zur L1-Norm 67
10.4 Transferfunktion des Filters $\vec{D}_{2x}$ 68
10.5 erste und zweite Ableitung von verrauschten Daten 68
10.6 Anwendung des Laplace-Filters auf ein Testbild 69
10.7 Transferfunktion des einfachen Laplace-Filters 69
10.8 Erste und Zweite Ableitung einer Gauß-Kurve. Man sieht die Ähnlichkeit zur idealen Transferfunktion von Gradienten- und Laplace-Operatoren. 71

11.1 Berechnung des Strukturtensors eines Beispielbildes 77
11.2 Regressionsgerade an die Fourier-Transformierte einer einfachen Nachbarschaft 78
11.3 Beispiel für lokalen Wellenzahlen und Phasen 79

12.1 verschiedene Texturen 83
12.2 Verschiedene Texturen mit dem selben Grauwerthistogramm 84
12.3 Mit Mittelwert und Varianz ununterscheidbare Histogramme 84
12.4 verschieden Bandpass-gefilterte Bilder einer Gardine (Original ganz links) 85

13.1 Projektion einer 3D-Bewegung (Geschwindigkeit $\vec{v}$) auf die Kamera-Ebene (optischer Fluss $\vec{f}$). Es zeigt sich, dass verschiedene Bewegungen $\vec{v}$ und $\vec{v}'$ auf den selben optischen Fluss $\vec{f}$ führen. 86
13.2 Illustration zum Blendenproblem, nach [Jähne 2005] 86
13.3 Bewegungsdetektion durch Differenzbildung. Die Differenzbilder sind kontrastverstärkt. Das linke Differenzbild zeigt den Effekt einer Beleuchtungsänderung. Im rechten Bild wurde der Wecker verschoben. 87
13.4 Object-Tracking: Es werden zwei Bilder einer Sequenz verglichen. Rechts sind die Bilder überlagert und die Bewegungsvektoren sind eingezeichnet (optischer Fluss $\vec{f}$). Man erkennt, dass die Methode nicht eindeutig ist (rechts unten!) 88
13.5 Bewegung in Raum und Zeit für eine 1D und eine 2D Bildsequenz. Die dicke grüne Linie bezeichnet jeweils die Bewegung. Auf den flanken der 2D-Sequenz sind Schnitte durch das Bild aufgezeichnet. 88
13.6 Veranschaulichung zur Kontinuitätsgleichung 91

14.1 Van-Critter-Iteration auf einem weich-gezeichneten Testbild 96

15.1 bimodale Histogramme zweier Bilder 98
15.2 Segmentierung eines Bildes mit dem Schwellenwertverfahren mit verschiedenen Schwellenwerten s. 99
15.3 Größen-Fehler beim Segmentieren bei verschieden hellen Objekten 99
15.4 Festlegung des Schwellenwertes s bei linearen und nicht-linearen Kanten 100
15.5 Ununterscheidbare Objekte beim kantenbasierten Segmentieren 100
15.6 Modellierung einer Kante im Bild mit Verschmierung und Beleuchtung 101
15.7 Veranschaulichung des pyramid-linking-Verfahrens 102

15.8 falsches Modell (Bild 1) für die Daten (Bild 2) 103
15.9 Beispiel zur 1D-Hough-Transformation 104
15.10 Verschiedene Versuche eine Funktion f an die messdaten zu fitten. rechts ist das Fehlerfunktional gezeigt. Es muss das Minimum aufgesucht werden. 104

16.1 Anwendung von Dilatation und Erosion auf Testbilder 108
16.2 Öffnen und Schließen mit einer 3×3-Maske auf ein Testbild angewendet. 110

17.1 Trägheitstensor eines Objektes 113
17.2 Kettencodierung eines Objektes mit verschiedenen Nachbarschaften 114
17.3 äquidistante Abtastung einer 2D-Kurve (rot) mit $P = 21$ Punkten 114
17.4 Flächenberechnung mit Kettencode 116

18.1 Merkmalsraum für ein einfaches Klassifikationsbeispiel von einem Bild mit Reis- und Pfefferkörnern 117
18.2 Merkmalsraum mit korellierten Merkmalen 118
18.3 Merkmalsraum mit korrelierten Merkmalen nach der Hauptachsentransformation 119

19.1 Komplexe Zahlenebene 122

A Weblinks

- Canny-Filter:
  http://en.wikipedia.org/wiki/Canny
- Gauß- und Laplace-Pyramide:
  http://www.cis.udel.edu/~qili/ta/cis489/2/index.html
- Hilbert-Transformation :
  http://en.wikipedia.org/wiki/Hilbert_transform
- Hough-Transformation:
  http://www.cs.tu-bs.de/rob/lehre/bv/Hough.html
- MatLab:
  http://www.prip.tuwien.ac.at/Teaching/SS/EFMEN/links.html
- Segmentierung:
  http://www.mathematik.uni-ulm.de/stochastik/lehre/ws05_06/seminar/sauter.pdf
-

Literaturverzeichnis

[Boas 1983] *Boas, Mary L. (1983):* **Mathematical Methods In The Physical Science**, 2. Auflage, New York - Brisbane - Toronto - Singapore: Wiley.

[Jähne 2005] *Jähne, Bernd (2005):* **Digitale Bildverarbietung**, 6. Auflage, New York - Berlin - Heidelberg: Springer.

[Haußecker, Spieß 1999] Haußecker, H, Spies, H (1999): **Chapter 13 Motion**, in: Handbook of Computer Vision and Applications, Volume 2 (Signal Processing and Pattern Recognition), Academic Press

[Kahder etal. 2001] Kahder, M.H., Basser, P.j., Parker, D.L., Alesander, A.L. (2001): **Analytical Computation of Eigenvalues and Eigenvectors in DT-MRI**, in: Journal of Magnetic Resonance **152**, S. 41-47

Index

δ-Kamm, 31
γ-Transformation, 39
γ-Wert, 10
γ-Korrektur, 10, 39
Ähnlichkeitsbedingung, 105
Öffnen, 109
Überabtastung, 32
überwachte Klassifizierung, 119
2D-DFT, 21

Abtast-Theorem, 50
Abtastpunkt, 29
Abtasttheorem, 20, 29
Abtastung/Digitalisierung, 28
Aliasing, 30
Amplitude, 14
analytisches Signal, 80
anisotrope Diffusion, 106
Assoziativität, 45

bandbegrenztes Signal, 30
Bandpass, 27, 50, 51, 94
Betragsberechnung, 66
Bewegungserkennung, 86
Bewegungsfeld, 90
Bilderzeugung, 28
Bildmittelung, 39
Bildpunkt, 9
bimodales Histogramm, 98
Binärbild, 98, 107
Binomialfilter, 51, 58, 71
Binomialkoeffizienten, 58
Binomialverteilung, 59
Blendenproblem, 86, 91

Canny-Filter, 71
Canny-Kantendetektion, 71
CCD-Kamera, 29
Closing, 109
CMYK, 10
Cosinus (cos), 13
Cosinus-Transformation, 27

darkfield, 40
Delta-Kamm, 31
DFT, 19
Difference-of-Gaussian-Filter, 72
Diffusionsgleichung, 53
Digitalkamera, 29
Dilatation, 41, 107
diskrete Differenzen, 67
Diskrete Fourier-Transformation, 19
Diskretisierungsgitter, 29
Distributivität, 45, 109
DoG-Filter, 72
Dreieckgitter, 11
Dualität, 109
Dunkelbild, 40
dyadische Punktoperation, 37
dynamischer Bereich, 20

Ecke, 64
einfache Nachbarschaft, 73
einstellbare Mittelung, 62
Entscheidungsraum, 119
Erosion, 107
Euler-Lagrange-Gleichungen, 105
Expansionsoperator, 51
exponentieller Skalenraum, 54

Faltung, 42, 63
Faltungstheorem, 25, 32
Farbbild, 10
Farbsystem, 10
Fast-Fourier-Transformation, 27
Fehlerfunktional, 104
Fensterfunktion, 31, 41
FFT, 27
Fick'sches Gesetz, 53
Finite-Impulse-Response-Filtern, 45
FIR-Filter, 45
Fläche, 116
Formparameter, 115
Fourier-Darstellung, 16
Fourier-Entwicklung, 16
Fourier-Reihe, 16
Fourierdeskriptoren, 114
Frequenz, 13

Gaborfilter, 82
Gamma-Korrektur, 10
Gauß'sches Rauschen, 60
Gauß-Filter, 51, 53
Gauß-Pyramide, 50, 102
Gaußfilter, 71
geometrische Transformationen, 41
gerade Symmetrie, 43
gespiegelter Rand, 44
gewichtete Mittelung, 62
Gitter, 11
Glättungsfilter, 56
Glattheit, 25
Glattheitsbedingung, 105
Gradientenfilter, 71

Grauwert, 9

Haartransformation, 27
Hadamardtransformation, 27
Hartley-Transformation, 27
Hauptachsentransformation, 118, 120
Hilbert-Filter, 80
Hilbert-Transformation, 80
Hilbertfilter, 79, 94
Hilberttransformation, 79
Hit-Miss-Operator, 110
homogene Koordinaten, 41
homogene Punktoperation, 37
Horner-Schema, 96
Hough-Transformation, 103
HSB, 10

IIR-Filter, 48
Infinite-Impulse-Response-Filter, 48
inhomogene Diffusion, 106
inverse Filterung, 95

Kante, 64
kantenbasierte Segmentierung, 100
Kantendetektion, 64
kausaler Filter, 48
Kern, 26
Klassifikation, 117
Klassifikator, 119
Kluster, 117
Kohärenzmaß, 76
Kommutativität, 45, 109
Kompaktheit, 25
Komplexe Zahl, 122
Komplexe Zahlenebene, 122
Kontinuierliche Modellierung, 104
Kontinuitätsgleichung, 53
Korrelationsmethode, 93
Kovarianzmatrix, 118
Kreuzkorrelationskoeffizient, 94
Kreuzkovarianz, 118

Lämgenskala, 50
Lagrange-Funktion, 105
Lagrange-Gleichungen, 105
Laplace-of-Gaussian-Filter, 72
Laplace-Operator, 69
Laplace-Pyramide, 50, 51, 94
Laplacepyramide, 85
laufender Mittelwert, 57
Lauflängenkodierung, 34
Least-Square-Fit, 70, 104
Lempel-Ziv-Welch-Algorithmus, 35
lineare Spreizung, 38
lineare verschiebungsinvariante Filter, 42
Linearität, 44
Linie, 64
LoG-Filter, 72
Lokale Fourier-Transformation, 27
lokale Nachbarschaft, 90
lokale Phase, 78, 94
lokale Wellenzahl, 78
Lookup-Table, 37
LSI, 42
LUT, 37, 38
LZW-Algorithmus, 35

Medianfilter, 62
Mehrdimensionale Fourier-Transformation, 21
Mehrschrittmittelung, 61
Merkmalsraum, 117
Merkmalsvektor, 117
Methode der höchsten Wahrscheinlichkeit, 120
Methode des geringsten Abstandes, 120
Minimum-Maximum-Prinzip, 54
Mittelung, 56
Mittelwert, 84
Modell, 103
modellbasierte Segmentierung, 103
Modellierung, 103
Moduloarithmetik, 10
Moiré-Mustern, 30
Monotonie, 109
Morphologie, 107
morphologische Operation, 107
Multiskalenrepräsentation, 50

Nachbarschaft, 73
Nachbarschaftsbeziehungen, 11
Nachbarschaftsoperation, 42
Nachschaumethode, 120
Nagelbrettfunktion, 31
Newton-Raphson-Verfahren, 94
nicht-lineare Störungen, 40
nichtlineare und steuerbare Mittelung, 62
normalisierte Faltung, 63
nullphasiger Filter, 56

Object-Tracking, 87
octree, 34
Opening, 109
Operatornotation, 42
optischer Fluss, 86, 90
Orientierung, 73
Ortsraum, 50

Parameterraum, 103
Phase, 14
Phasenmethode, 94
Phasenschieber, 80
Phasenverschiebung, 65
Pixel, 9
Pixelgitter, 11
pixelorientierte Segmentierung, 98
polare Fourierdeskriptoren, 115
Polarform, 122
Punktantwort, 45, 48
Punktoperation, 37
Pyramid-Linking, 102

Quadermethode, 120
quadratintegrable Funktion, 18

quadratischer Skalenraum, 54
Quadraturfilter, 94
quadtree, 34
Quantisierung, 28

räumliche Frequenz, 13
räumliche Informationen, 50
Rückfaltung, 95
radiometrische Zweipunkt-Korrektur, 40
Randeffekte bei Faltungen, 43
Rangordnungsfilter, 47
Reaktions-Diffusions-System, 105
Rechteckfilter, 57
Rechteckgitter, 11
Region-Growing, 101
Regionenbild, 98
regionenorientierte Segmentierung, 101
regularisierter Kantendetektor, 70
Rekursive Mittelung, 61
rekursives Filter, 48, 61
RGB, 10
Richtung, 73
Richtungsketten, 113
Richtungszerlegung, 94
Richtungszerlegungen, 52
RLE, 34
Rotation, 41
run length encoding, 34
Rundheit, 116

Sättigungsarithmetik, 10
Salt-and-Pepper-Rauschen, 60
Scherung, 41
Schließen, 109
Schwarz-Weiß-Bild, 34, 113
Schwebung, 20
Schwellenwert-Verfahren (Segmentierung), 98
Secheckgitter, 11
Segmentierung, 98
selbstlernende Klassifizierung, 119
Separabilität, 24
Sinus (sin), 13
Sinus-Fenster, 41
Sinus-Transformation, 27
Skala, 50
Skalenparameter, 53
Skalenraumtheorie, 52
Sobel-Filter, 71
Split-and-Merge, 101
stabiler Filter, 49
Standardabtastung, 32
Stauchung, 41
Strukturelement, 108
Superpositionsprinzip, 108
Symmetrie, 65

Textur, 83
Tiefpass, 51
Tiefpassfilter, 56
Trägheitsellipsoiden, 113
Trägheitstensor, 78
Trainingsdatensatz, 119
Transferfunktion, 46, 49
Translation, 41

Umfang, 116
unüberwachte Klassifizierung, 119
ungerade Symmetrie, 43
unitäre Transformation, 26

Van Critter Iteration, 96
Varianz, 62, 84, 118
Varianzbild, 62
Variationsmethode, 104
Verschiebung, 24
Verschiebungsfreiheit, 56, 65
Verschiebungsinvarianz, 45, 108
Verschiebungsoperator, 45
Vollständigkeitsrelation, 16

Wellenfront, 15
Wellenlänge, 13
Wellenvektor, 15, 21
Wellenzahl, 13
Wichtungsbild, 63

z-Transformation, 49
Zahlenbereich, 9
zyklische Faltung, 44
zyklische Fortsetzung, 44