Signaltheorie in der Bildverarbeitung

Wolfgang Ortmann

Jena, 2021

Inhaltsverzeichnis

Teil I.

Einführung

1. Signale und Funktionen

1.1. Signale

Gegenstand der Signaltheorie sind Signale. Signale sind zeit- oder ortsabhängige physikalische Größen. Dabei wird diese Größe und ihr Verlauf als Nachricht oder allgemein als Information betrachtet.

Durchläuft ein Signal ein System - eine Übertragungskette oder eine Verarbeitungseinrichtung - wird es verändert. Die Signaltheorie befasst sich mit der Analyse der Veränderungen, ihrer Korrektur und Verfahren zur Extraktion von Informationen.

Die Signaltheorie beschreibt Signale in der Form von Funktionen. Ging es vor allem mit dem Hintergrund der analogen Nachrichtenübertragung zunächst um kontinuierliche zeitabhängige Signale, kamen mit der Betrachtung ortsabhängiger Signale auch mehrdimensionale Funktionen mehrerer Veränderlicher dazu. Die Verwendung der Digitaltechnik führte zu einer steigenden Bedeutung diskreter Funktionen.

1.2. Funktionen

Signale sind physikalische Größen, ihre Beschreibung erfolgt durch Funktionen. Aus einzelnen Gebieten seien hier einige Beispiele genannt:

Akustik: Schall stellt eine Funktion des Luftdruckes über die Zeit dar ($p(t)$). Die Ausbreitung des Schalls im Raum lässt daraus eine Funktion in Zeit und dreidimensionalen Raum werden ($p(x, y, z, t)$).

Bei einer schwingenden Saite ist die Auslenkung y zu jedem einzelnen Zeitpunkt eine Funktion des Ortes $y(x)$. Betrachtet man den zusätzlich zeitlichen Verlauf, so erhält man eine Funktion des Ortes und der Zeit $y(x, t)$.

Elektroakustik: Elektrische Spannungen, die den Schall übertragen, sind ebenfalls eine Funktion der Zeit $U(t)$.

Elektronische Nachrichtentechnik: Elektromagnetische Schwingungen und Wellen $U(t)$ bzw. $U(x, y, z, t)$

Optik: Bilder sind zweidimensionale Funktionen. Sie beschreiben einen Wert wie Helligkeit, Intensität des Lichtes, Reflexionsvermögen oder Farbwerte in Abhängigkeit vom Ort in der Ebene.

Wenn wir über Funktionen reden, müssen wir einige grundlegende Arten von Funktionen

unterscheiden. Wir wollen dies als “Bildmodell” bezeichnen, da wir uns hier vorrangig mit Bildern beschäftigen wollen, aber die entsprechende Unterscheidung gilt natürlich auch für andere Signale.

1.2.1. Dimensionalität

Bilder sind meist zweidimensional. Als dritte Dimension kommt bei Bewegtbildern die Zeit dazu. In 3D-Bildern ist die z-Achse die dritte Dimension. Eindimensionale Bilder gibt es eigentlich nicht. Eine ganze Reihe von Anwendungen auch in der Bildverarbeitung wird jedoch auch mit eindimensionalen Funktionen umgehen. Für ein allgemeines Bildmodell ist deshalb der Ansatz einer beliebigen Zahl von Dimensionen sinnvoll.

Bei der Darstellung mathematischer Zusammenhänge werden wir vieles zunächst für eindimensionale Funktionen betrachten, weil es dadurch leichter darstellbar ist. Meist ist die Erweiterung auf höhere Dimensionen einfach. In allen wichtigen Fällen werden wir natürlich auch die zweidimensionale Form explizit angeben. Das gilt erst recht dann, wenn die Übertragung auf das Zweidimensionale nicht offensichtlich ist.

1.2.2. Analog oder Digital

Die meisten Ausgangs-Signale sind reellwertige Funktionen, die über die reellen Zahlen definiert sind. In Analogie zu den technischen Begriffen wollen wir diese als **analog** bezeichnen. Vor allem durch die Verarbeitung mittels digitaler Technik werden diskrete Funktionen wichtiger: Die analogen Signale werden abgetastet, wodurch sich der Definitionsbereich auf diskrete Stützstellen reduziert (Diskretisierung). Die resultierenden Funktionen wollen wir als **digital** oder auch diskret bezeichnen.
Außerdem wird der Funktionswert digitalisiert, was eine Quantisierung der möglichen Werte zu Folge hat. Die Quantisierung der Funktionswerte wirkt sich auf die Genauigkeit aus: Der analoge Wert wird auf einen der in einem Raster liegenden Werte gerundet und damit verfälscht. Die Folge ist ein Quantisierungsfehler, der durch technischen Aufwand und kleinere Quantisierungsstufen klein gehalten werden kann.

Die Diskretisierung hat grundlegendere Auswirkungen. Von den überabzählbar vielen Funktionswerten der analogen Funktion werden abzählbar viele abgetastet. Eine wichtige Frage wird sein, ob man erreichen kann, dass dabei keine Information verloren geht, was uns später zum Abtasttheorem führen wird.

Diskrete Funktionen sind aber nicht immer nur eine Näherung der originalen analogen Funktion. Es gibt auch in der realen Welt diskrete Funktionen: Ein Kristallgitter beschreibt eine Anordnung von Atomen in einem Festkörper. Wellen, die sich in diesem Gitter ausbreiten, sind eine diskrete Funktion der Auslenkung über die Stellen des Gitters. Aber auch dann, wenn die Diskretisierung einfach technisch bedingt ist, weil wir den Computer zur Verarbeitung verwenden wollen, ist es natürlich interessant, wie sich diese diskreten Funktionen verhalten und was man mit ihnen anstellen kann.

Wir wollen analoge und digitale Funktionen auch in der Notation unterscheiden. Sei zum Beispiel f eine analoge Funktion, so bezeichnen wir den Funktionswert an der Stelle x mit $f(x)$. Ist f hingegen eine digitale Funktion, ist der Funktionswert an einer Stelle i die Auswahl eines von abzählbar vielen Elementen und wir schreiben f_i. Damit drücken wir auch die Verwandschaft digitaler Funktionen mit Folgen und Vektoren aus. Das Argument ist hier immer ein Index, also eine ganze Zahl. Erzeugen wir eine digitale Funktion f' durch Abtastung einer analogen Funktion f, so sind die Funktionswerte der digitalen Funktion die Werte der analogen Funktion an den Stützstellen x_k.

$$f'_k = f(x_k)$$

Wir werden (wenn nicht anders vermerkt) immer von äquidistanten Stützstellen ausgehen, die Vielfache einer Schrittweite Δx sind:

$$f'_k = f(x_k) \quad \text{mit} \quad x_k = k \cdot \Delta x$$

1.2.3. Endlich oder unendlich

Signale haben typisch einen beschränkten endlichen Wertebereich. Bei einer konkreten Problemstellung werden bestimmte Schranken typischerweise nicht überschritten. Bei der technischen Verarbeitung sind die Schranken meist auch durch technische Grenzen vorgegeben.

Bei Bildern wird man auch spontan von einem endlichen Definitionsbereich ausgehen: alle Bilder, mit denen wir umgehen, sind endlich. Aber wir dürfen nicht vergessen, dass diese Endlichkeit eigentlich meist künstlich ist: die reale Welt endet nicht mit dem Rand unseres Bildes. Wir beschränken uns auf einen endlichen Ausschnitt der realen Welt, weil

- wir nicht genug Speicherplatz für unendlich viele Werte haben,
- uns nur dieser Bereich interessiert oder
- der begrenzte Ausschnitt künstlerisch wertvoll erscheint.

In jedem Fall werden wir uns mit endlichen und unendlichen Definitionsbereichen befassen müssen.

Ein unendlicher Definitionsbereich hätte für die Bildverarbeitung einen großen Vorteil: Wir brauchen keinen Rand des Bildes zu betrachten. Eine Verschiebung, Streckung oder Stauchung der Funktion wäre kein Problem, kein Teil der Funktion wandert aus dem Definitionsbereich heraus oder kommt von außen hinein. Jeder Punkt hat eine Umgebung. Der Nachteil ist aber, dass man Konvergenzprobleme betrachten muss, wenn man Summen oder Integrale berechnen will.

Bei einen endlichen Definitionsbereich haben wir kein Konvergenzproblem mehr. Dafür entstehen dann aber neue Probleme: Bei einem endlichen Definitionsbereich gibt es einen Rand. Bei geometrischen Transformationen wie Verschiebung, Streckung oder Stauchung der Funktion wandern Bildpunkte in das Bild hinein oder hinaus. Andere Operationen, wie zum Beispiel die Faltung, erfordern die Betrachtung einer Umgebung eines Pixels, so dass

man für Pixel am Rand über den Rand hinaus zugreifen müsste. Man benötigt also eine Definition, was dort vorzufinden ist. In speziellen Fällen kann man davon ausgehen, dass die Werte außerhalb den Wert Null haben. Dann ist es genau genommen kein endlicher Definitionsbereich, sondern wir haben eine Funktion mit beschränktem Träger und unendlichem Definitionsbereich. Die Werte außerhalb als Null anzunehmen, ist aber natürlich immer **eine** Möglichkeit der Fortsetzung. Man nennt dies zero padding. Eine andere Möglichkeit ist, eine periodische Fortsetzung der Funktion anzunehmen. Natürlich gibt es noch viele weitere Möglichkeit, wie zum Beispiel die gespiegelte Fortsetzung. Wenn jedoch nichts über die Funktion bekannt ist, ist jede Annahme mehr oder wenig willkürlich. Dann spricht aber auch nichts dagegen, die Fortsetzung auszuwählen, die sich besonders gut handhaben lässt: Wir werden meist von einer periodischen Fortsetzung ausgehen - unsere Funktion ist periodisch und es gilt

$$f(x) = f(x + X)$$
$$f_n = f_{n+N}$$

Diese Betrachtungsweise bringt einige Vorteile mit sich:

- Wir können den betrachteten Definitionsbereich beliebig verschieben und finden immer wieder die gleichen Werte vor. Wir werden oft Intervalle von $0 \leq n < N$ beziehungsweise $0 \leq x < X$ verwenden. Dies ist keine nennenswerte Einschränkung: ein Übergang zu einem anderen Bereich, zum Beispiel symmetrisch um die Null, ist jederzeit möglich. So ist $f_{-1} = f_{N-1}$ usw.
- Wenn wir über den gesamten Definitionsbereich integrieren oder summieren müssen, können wir die Intervallgrenzen beliebig verschieben, ohne den Wert zu ändern.

1.2.4. Bildmodelle

Wir müssen im folgenden also mit verschiedenen Arten von Funktionen, Bildmodellen, umgehen. Viele Eigenschaften und Zusammenhänge lassen sich von einem Modell auf das andere übertragen. Es ist aber immer eine genaue Unterscheidung notwendig. Wir wollen dies wie folgt bezeichnen:

- A1$[X]$, A2$[X, Y]$
  ein- und zweidimensionale analoge Funktionen mit einer Periode von X und Y
  Meist werden wir diese als Funktion über dem Intervall $0 \leq x < X$ bzw. $0 \leq x < X, 0 \leq y < Y$ betrachten.
- A1$[\infty]$, A2$[\infty, \infty]$
  ein- und zweidimensionale analoge Funktionen von minus Unendlich bis plus Unendlich
- D1$[N]$, D2$[M, N]$
  ein- und zweidimensionale diskrete Funktionen mit einer Periode N bzw. M, N
- D1$[\infty]$, D2$[\infty, \infty]$
  ein- und zweidimensionale diskrete Funktionen von minus Unendlich bis plus Unendlich

1.3. Wichtige Funktionen

- Sinusförmige Funktionen sin und cos
  Als sinusförmige Funktionen wollen wir die Funktionen sin und cos in allen Kombinationen $\lambda_1 \cdot \sin(k \cdot x) + \lambda_2 \cos(k \cdot x)$ bezeichnen, was gegenüber dem "reinen" Sinus eine Streckung oder Stauchung bezüglich der Achsen und eine Verschiebung darstellt.
  Die Eulersche Formel
  $$e^{i\varphi} \;=\; \cos\varphi \;+\; i\sin\varphi$$
  erlaubt eine Darstellung der entsprechenden komplexen Funktionen als $\lambda e^{ik(z+\varphi)}$.
  Wir werden später meist die komplexe Darstellung verwenden, auch wenn unsere Signale, also die Bilder, meist reellwertig sind. Der Vorteil der komplexen Darstellung mittels der e-Funktion ist das einfache Rechnen. Das Additionstheorem lautet hier zum Beispiel
  $$e^{i(\varphi_1+\varphi_2)} = e^{i\varphi_1} \cdot e^{i\varphi_2},$$
  was sich einfach aus den Potenzgesetzen ergibt. Aufgelöst nach sin und cos ergibt sich
  $$\cos(\varphi_1+\varphi_2) + i\sin(\varphi_1+\varphi_2) = (\cos(\varphi_1) + i\sin(\varphi_1)) \cdot (\cos(\varphi_2) + i\sin(\varphi_2))$$
  was nach Ausmultiplizieren und getrennter Betrachtung von Real- und Imaginärteil die Additionstheoreme für sin und cos ergibt:
  $$\begin{aligned} \cos(\varphi_1+\varphi_2) &= \cos(\varphi_1)\cdot\cos(\varphi_2) - \sin(\varphi_1)\cdot\sin(\varphi_2) \\ \sin(\varphi_1+\varphi_2) &= \cos(\varphi_1)\cdot\sin(\varphi_2) + \sin(\varphi_1)\cdot\cos(\varphi_2) \end{aligned}$$
  Offensichtlich ist hier die Rechnung mit der e-Funktion und den Potenzgesetzen einfacher.
- Die Funktion atan2 oder $\arctan 2$
  Der Name der Funktion atan2 leitet sich vom Arkustangens ab. Bei der Umrechnung von kartesischen Koordinaten (x, y) in Polarkoordinaten (r, φ) muss der Winkel φ berechnet werden. Aus
  $$\begin{aligned} x &= r \cdot \cos(\varphi) \\ y &= r \cdot \sin(\varphi) \end{aligned}$$
  ließe sich durch dividieren einfach ableiten
  $$\tan(\varphi) = \frac{y}{x}$$
  und damit
  $$\varphi = \arctan(\frac{y}{x}) + k\pi$$

Wenn man sich diese Lösung ansieht, stellt man eine Mehrdeutigkeit fest: Die Periode des Tangens ist π oder 180°, nicht 2π oder 360°, wie zu erwarten. Der Fehler dieser Rechnung ist die Division von y durch x : x, y und $-x, -y$ ergeben das gleiche Ergebnis. Man könnte jetzt entsprechende Fallunterscheidungen einbauen. Besser ist die Verwendung der Funktion atan2, die als Argumente die Werte von y und x erwartet und den Winkel φ zurückgibt.

$$\varphi = \text{atan2}(y, x) + 2k\pi$$

Das Ergebnis ist identisch zur Anwendung der Argument-Funktion $\arg(z)$ auf die komplexe Zahl $z = x + iy$.
Vorsicht: Die Reihenfolge der Argumente wird in der Literatur gelegentlich vertauscht!

- Die sinc-Funktion
  Diese Funktion ist definiert zu

  $$\text{sinc}(x) \;=\; \text{si}(x) \;=\; \frac{\sin(x)}{x}$$

  Sie wird auch oft als *Spaltfunktion* bezeichnet, weil sie die Lichtamplitude bei der Beugung von Licht an einem Spalt beschreibt. Als "normierte" Form wird manchmal

  $$\text{sinc}(x) \;=\; \frac{\sin(\pi x)}{\pi x}$$

  definiert. Eine Unterscheidung der Bezeichnungen von *sinc* für die normierte Form und *si* für die nicht normierte Form ist gelegentlich anzutreffen, wird aber nicht einheitlich verwendet. Wir werden *sinc* für die erste Definition der (nicht normierten) sinc-Funktion verwenden.
- Der Einheitsimpuls δ
  Sowohl der diskrete Einheitsimpuls als auch der analoge Einheitsimpuls (**Dirac-Funktion**) spielen eine große Rolle. Der diskrete Einheitsimpuls ist leicht zu verstehen:

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

  Der analoge Einheitsimpuls, die **Dirac-Funktion**, ist keine normale Funktion mehr. Es ist eine verallgemeinerte Funktion oder **Distribution**. Für alle Argumente $x \neq 0$ ist der Funktionswert Null, für $x = 0$ ist der Funktionswert Unendlich.

  $$\int_B \delta(\xi) d\xi = \begin{cases} 1 & ; 0 \in B \\ 0 & ; 0 \notin B \end{cases}$$

- Die Rechteckfunktion $\text{rect}(x)$
  Die Rechteckfunktion ist ein typischer Verteter einer unstetigen Funktion:

  $$\text{rect}(x) \;=\; \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

- Die Gauß-Funktion
  Die mehrdimensionale Gauß-Funktion lautet:

$$G(x;\mu,\Sigma) \;=\; \frac{1}{2\pi\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

  Dabei ist μ der Erwartungswert-Vektor und Σ die auf den Erwartungswert-Vektor zentrierte Kovarianz-Matrix.

1.4. Elementare Operatoren

Man kann sicher zahlreiche Operatoren definieren, die angewendet auf eine Funktion eine neue Funktion erzeugen. Wir wollen zunächst zwei elementare Operatoren definieren:

- Verschiebungsoperator $S_{\Delta x}$ (Shift)
  Die Anwendung des Verschiebungsoperators verschiebt alle Funktionswerte einer Funktion f um einen Betrag Δx. Für die verschobene Funktion $f_s = S_{\Delta x} f$ gilt also

$$f_s(x+\Delta x) = (S_{\Delta x} f)(x+\Delta x) = f(x)$$

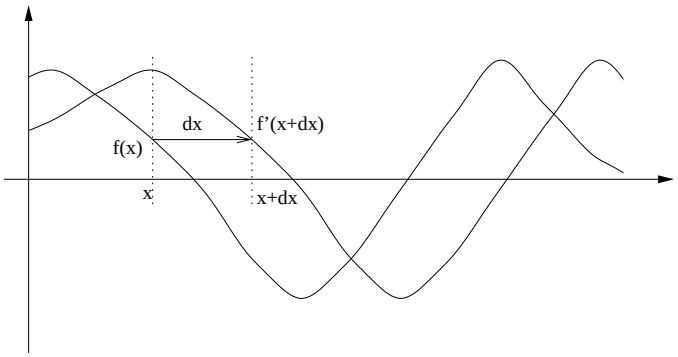


Abbildung 1.1.

  Für eine sinnvolle Berechnung der verschobenen Funktion formulieren wir um zu

$$f_s(x) = (S_{\Delta x} f)(x) = f(x-\Delta x)$$

  Für digitale Funktionen schreiben wir entsprechend:

$$f_n^s = (S_{\Delta n} f)_n = f_{n-\Delta n}$$

- Spiegelungsoperator R (Reflection)
  Der Spiegelungsoperator spiegelt die Funktion am Ursprung. Es gilt für analoge Funktionen

$$(Rf)(x) = f(-x)$$

und für digitale Funktionen

$$(Rf)_n = f_{-n}$$

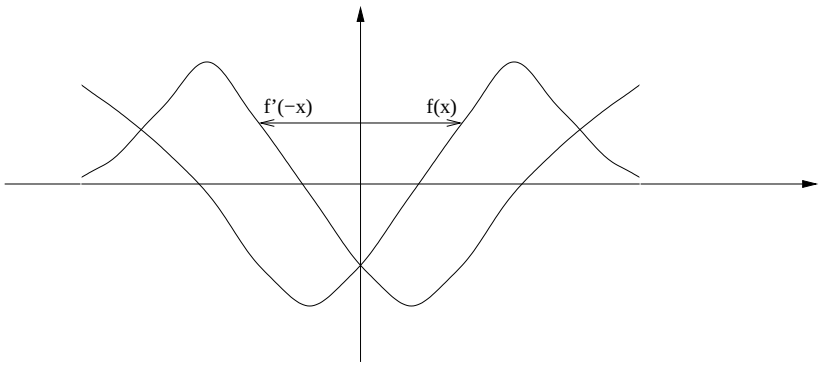


Abbildung 1.2.

Für eine vereinfachte Schreibweise wollen wir für die gespiegelte Funktion f^R schreiben, so dass gilt

$$f^R(x) = (Rf)(x) = f(-x)$$

und

$$f_n^R = (Rf)_n = f_{-n}$$

Teil II.

Faltung und Korrelation

2. Faltung und Korrelation

2.1. Faltung

Die Faltung ist eine der wichtigsten Operationen in der Signalverarbeitung (lineare Systemtheorie) und der Bildverarbeitung. In der englischsprachigen Literatur heißt diese **convolution**, manchmal auch **folding**. Für diskrete Funktionen ist die Faltung * definiert zu:

$$(f * g)_n = \sum_{i=-\infty}^{\infty} f_i g_{n-i} \tag{2.1}$$

Für analoge Funktionen gilt entsprechend:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\xi) g(t-\xi) d\xi \tag{2.2}$$

Die Erweiterung in das zweidimensionale erfordert zwei Summen oder Integrale:

$$(f * g)_{m,n} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_{i,j} g_{m-i,n-j} \tag{2.3}$$

und

$$(f * g)(t_1, t_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\xi, \eta) g(t_1 - \xi, t_2 - \eta) d\xi d\eta \tag{2.4}$$

Hier wird zunächst ein unendlicher Definitionsbereich der beiden Funktionen vorausgesetzt und darüber integriert beziehungsweise summiert.

Das Ergebnis der Faltung zweier Funktionen ist wieder eine Funktion.

Häufig findet man in der Literatur eine falsche Schreibweise:

$$f(t) * g(t) = \int_B f(\xi) g(t-\xi) d\xi$$

Das ist nicht korrekt: gefaltet werden nicht die zwei Funktionswerte $f(t)$ und $g(t)$, sondern die zwei Funktionen f und g. Anschließend wird das Ergebnis an der Stelle t betrachtet. Stellvertretend für alle Definitionen werden wir oft die eindimensionale, diskrete Version zur Darstellung verwenden.

Für ein besseres Verständnis, was sich hinter der Faltung verbirgt, schreiben wir die diskrete Faltung nochmals auf und verwenden den Verschiebungsoperator:

$$(f * g)_n = \sum_i f_i \cdot g_{n-i} = \sum_i f_i \cdot (S_i g)_n$$

oder auch

$$f * g = \sum_i f_i \cdot S_i g$$

Wir nehmen also die Funktion g und verschieben sie um alle möglichen Werte i. Die jeweiligen verschobenen Funktionen wichten wir mit f_i und addieren diese. Anders gesagt, die Funktion g beschreibt, wie sich ein Funktionswert von f "auf die Umgebung verteilt".

Die Faltung besitzt alle schönen mathematischen Eigenschaften - sie ist linear, kommutativ, assoziativ.

Recht leicht können wir die Kommutativität zeigen:

$$(f * g)_n = \sum_i f_i g_{n-i} = \sum_{n-l} f_{n-l} g_l$$

Nach der Substitution $i \to n-l$ benutzen wir eine oft benötigte Erkenntnis: Auch wenn wir in der Summe als Index jetzt $n-l$ stehen haben, wird immer über den gesamten Definitionsbereich summiert. Auch die geänderte Reihenfolge, der Index fällt jetzt bei steigendem l, ändert nichts am Ergebnis, da über alle Werte summiert wird. Deshalb dürfen wir $n-l$ einfach durch l ersetzen, ohne dass sich an der Summe etwas ändert.

$$(f * g)_n = \sum_{n-l} f_{n-l} g_l = \sum_l g_l f_{n-l} = (g * f)_n$$

Mit der Kommutativität ist klar, dass die Linearität bezüglich beider Operanden gelten muss:

$$(\lambda \cdot f + \mu \cdot g) * h = \lambda \cdot f * g + \mu \cdot g * h$$
$$f * (\lambda \cdot g + \mu \cdot h) = \lambda \cdot f * g + \mu \cdot f * h$$

Die anderen Eigenschaften lassen sich ebenfalls einfach zeigen.

Die Faltung hat viele interessante Anwendungen in der Bildverarbeitung, wird aber auch in anderen Gebieten benutzt. Zum besseren Verständnis zunächst eine einfache Anwendung in der Wahrscheinlichkeitsrechnung.

Beispiel 1: Gegeben seien zwei diskrete Zufallsvariablen X und Y. Der Einfachheit halber sollen diese nur ganze Zahlen annehmen. Ein typisches Beispiel dafür ist das Würfeln. Die Wahrscheinlichkeiten $P(X = i) = p_i$ und $P(Y = j) = q_j$ seien gegeben. Die Zufallsvariablen X und Y seien unabhängig. Gesucht ist nun die Verteilung der Summe der Zufallsvariablen, also $Z = X + Y$. Beim Würfeln würde das bedeuten, dass wir zweimal unabhängig voneinander würfeln und die Augenzahl addieren. Da die Zufallsvariablen

unabhängig sind, müssen wir die Wahrscheinlichkeiten aller Kombinationen von Werten addieren, die eine bestimmte Summe ergeben, also

$$P(Z=l) = \sum_{i,j,i+j=l} p_i \cdot q_j = \sum_i p_i q_{l-i} = (p*q)_l$$

Sind X und Y stetig verteilt und unabhängig mit den Dichten $f(x)$ und $g(y)$, dann ist die Dichte für die Summe ebenfalls

$$h(z) = (f*g)(z) = \int_{-\infty}^{\infty} f(x)g(z-x)dx$$

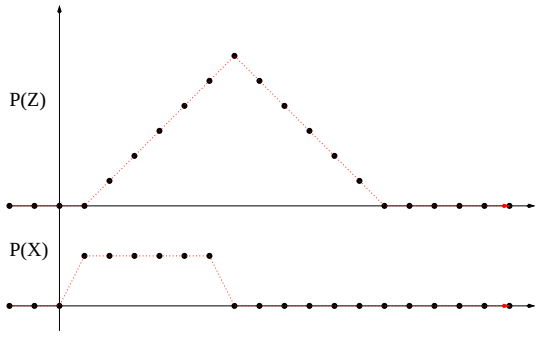
als Faltung beschrieben.

Aus der Wahrscheinlichkeitsrechnung ist bekannt: Sind X und Y normalverteilt, so ist auch Z normalverteilt. Damit können wir eine erste Eigenschaft der Faltung ablesen: Die Normalverteilung ist invariant gegenüber der Faltung: immer wenn man normalverteilte Dichten faltet, ist das Ergebnis die Dichte einer Normalverteilung.

Betrachten wir das Beispiel "Würfeln" mit einem klassischen Würfel. Für die Wahrscheinlichkeiten gilt

$$p_i = q_j = \begin{cases} 0 & i \leq 0 \\ \frac{1}{6} & 1 \leq i \leq 6 \\ 0 & i \geq 7 \end{cases}$$

Dies ist eine eindimensionale Funktion mit unendlicher Ausdehnung. Da die Funktionswerte ausserhalb des Intervalls $[1,6]$ Null sind, handelt es sich um eine Funktion mit beschränktem Träger. Nach der Faltung mit sich selbst dehnt sich der von Null verschiedene Bereich weiter aus, es gibt mehr als sechs Funktionswerte ungleich Null. Die Wahrscheinlichkeit ist eine Dreiecksfunktion mit $P(Z=7) = \sum_l p_l \cdot p_{7-l} = \frac{1}{6}$ als Maximum.



Beispiel 2: Gegeben seien zwei Polynome

$$p_1(x) = \sum_{i=0}^{n} a_i x^i \quad \text{und} \quad p_2(x) = \sum_{k=0}^{m} b_k x^k$$

Die Koeffizienten fassen wir als zwei diskrete Funktionen a und b auf, wobei $a_i = 0$ für $i < 0$ und $i > n$ und $b_k = 0$ für $k < 0$ und $k > m$ ist. Wir betrachten nun das Produktpolynom $p(x)$ und erhalten

$$p(x) = p_1(x) \cdot p_2(x) = \sum_{l=0}^{m+n} (a * b)_l \cdot x^l$$

Die Koeffizienten des Produktpolynomes ergeben sich als Faltung der Koeffizientenfunktionen. Dieses Beispiel hat sogar eine praktische Bedeutung. Wenn wir z.B. $x = 10$ setzen, dann stellt dies die Multiplikation zweier ganzer Zahlen in dezimaler Darstellung dar. Das Ergebnispolynom ist das Ergebnis der Multiplikation, allerdings noch ohne Behandlung des Übertrages. Wenn wir folglich auf einem Rechner eine schnelle Multiplikation zweier großer ganzer Zahlen implementieren wollen, können wir dies über die diskrete Faltung realisieren. Diese lässt sich mittels der Fouriertransformation schnell realisieren. Anschließend müssen wir nur noch die Überträge von Stelle zu Stelle durchreichen.

2.2. Faltungsoperator

Im wesentlichen eine andere Schreibweise der Faltung ist die Formulierung eines Faltungsoperators:

$$h = L_g f = g * f$$

Mit dieser Schreibweise als Operator drückt sich aber auch eine etwas andere Sicht aus: Der Operator L_g verarbeitet eine Funktion f und hat eine Funktion h als Ergebnis. g steuert als Parameter die Verarbeitung. Anders als bei diskreten Funktionen ist die tiefgestellte Schreibweise von g nicht als Index zu verstehen.

Jetzt lassen sich Eigenschaften des Faltungsoperators definieren. Aus der Linearität der Faltung leitet sich die Linearität des Faltungsoperators ab:

$$L_h(\lambda \cdot f + \mu \cdot g) = \lambda \cdot L_h f + \mu \cdot L_h g$$

2.3. Korrelation

Eng verwandt mit der Faltung ist die Korrelation $\circ$:

$$(f \circ g)_n = \sum_i f_{n+i}\overline{g_i} \quad \text{und} \quad (f \circ g)_{m,n} = \sum_i \sum_j f_{m+i,n+j}\overline{g_{i,j}}$$

bzw. für analoge Funktionen:

$$(f \circ g)(t) = \int_B f(t+\xi)\overline{g(\xi)}d\xi \quad \text{und} \quad (f \circ g)(t_1, t_2) = \int_{B_1}\int_{B_2} f(t_1+\xi, t_2+\eta)\overline{g(\xi,\eta)}d\xi d\eta$$

Überstreichungen bedeuten die Operation *konjugiert komplex*. Bei Funktionen verstehen wir darunter die elementweise Anwendung. Bei reellwertigen Funktionen kann diese Operation entfallen.

Die Korrelation lässt sich als Skalarprodukt der gegeneinader verschobenen Funktionen interpretieren. Dabei können wir f nach links oder g nach rechts verschieben:

$$(f \circ g)_n = \sum_i f_{n+i}\overline{g_i} = \sum_i f_n \overline{g_{i-n}} = \langle f, S_n g \rangle$$

Die Korrelation ist nicht identisch mit der Faltung, aber ist dieser sehr ähnlich. Leider ist sie jedoch weder kommutativ noch assoziativ. Statt der Kommutativität gilt:

$$(f \circ g)_m = \overline{(g \circ f)}_{-m} = \overline{((g \circ f)^R)}_m$$

Wir können weiter ablesen:

$$(f \circ g)_n = \sum_i f_{n+i}\overline{g_i} = \sum_l f_{n-l}\overline{g_{-l}} = \sum_l f_{n-l}\overline{g_l^R} = F * \overline{g^R}$$

Das ist aber die Faltung von f mit der konjugiert komplexen, am Koordinatenursprung gespiegelten Funktion g.

$$f \circ g = f * \overline{g^R} = f * (\overline{Rg})$$

Nun können wir viele Eigenschaften der Korrelation auf die Faltung zurückführen.

Für reellwertige Funktionen gilt

$$f \circ g = f * g^R = (g * f^R)^R = (g \circ f)^R$$

Für gerade Funktionen g gilt ja $g = g^R$. Damit ist die Korrelation gerader Funktionen f und g identisch zur Faltung und kommutativ

$$f \circ g = f * g^R = f * g = g * f^R = g \circ f \quad \text{mit f und g gerade}$$

2.4. Kreuzkorrelation und Autokorrelation

Eine Bedeutung der Korrelation besteht darin, die Ähnlichkeit zweier Funktionen bei einer gegenseitigen Verschiebung zu beschreiben. Dies war letztendlich namensgebend. Der Zahlenwert $(f \circ g)_n$ gibt an, wie gut f und g miteinander korrelieren, wenn man g um n verschiebt, oder, was dasselbe ist, wenn man f um $-n$ verschiebt. Das Matchingmaß ist das Skalarprodukt.

In diesem Zusammenhang wird die Korrelation zweier Funktionen oft als **Kreuzkorrelation** bezeichnet. Die Korrelation einer Funktion mit sich selbst wird im Gegensatz dazu als **Autokorrelation** bezeichnet.

Die Autokorrelation beschreibt, wie gut eine Funktion mit sich selbst korreliert, wenn man sie um n Positionen verschiebt. Trivialerweise muss die Autokorrelation bei $n = 0$ immer ihr Maximum haben, da f hier mit sich selbst am besten korreliert. Wir schreiben nochmals:

$$f \circ f = f * \overline{f}^R = \overline{f}^R * f$$

Für reelle Funktionen f ist die Autokorrelation eine **gerade** (symmetrische) Funktion

$$(f \circ f)^R = f^R \circ f^R = f^R * f = f * f^R = f \circ f$$

Interessant ist das Verhalten der Autokorrelation bei Verschiebung der Funktion: Die Autokorrelation ist **verschiebungsinvariant** und ändert sich nicht bei Verschiebung der Funktion:

$$(S_m f \circ S_m f) = \sum_i f_{n+i-m} \overline{f}_{i-m} = \sum_l f_{n+l} \overline{f}_l = (f \circ f)_n$$

2.5. Faltung und Korrelationen in endlichen Modellen

Wenn die zu faltenden Funktionen nur in einem endlichen Teilbereich Funktionswerte ungleich Null besitzen (Funktionen mit kompakten Träger), kann man die Summation beziehungsweise Integration auf einen Teilbereich einschränken. Sei $f_k = 0$ für $k < k_1$ und $k > k_2$ und $g_m = 0$ für $m < m_1$ und $m > m_2$. Bei der Berechnung der Faltung

$$(f * g)_n \;=\; \sum_{i=-\infty}^{\infty} f_i g_{n-i}$$

sind dann nur die Terme $f_i g_{n-i}$ ungleich Null, für die gilt $k_1 \leq i \leq k_2$ und $m_1 \leq n-i \leq m_2$. Die Summation muss somit nur für alle i mit $\max(k_1, -m_2 + n) \leq i \leq \min(k2, -m_1 + n)$ erfolgen. Das Intervall für i wird leer und somit der Wert der Faltung Null, wenn die untere Grenze die obere überschreitet. Werte ungleich Null können somit nur für $k_1 + m_1 \leq n \leq k_2 + m_2$ auftreten, auch das Ergebnis der Faltung ist eine Funktion mit kompaktem Träger.

Um Funktionen mit einem endlichen Definitionsbereich behandeln zu können, könnten wir also den Definitionsbereich ins Unendliche erweitern und die undefinierten Werte mit Null auffüllen. Diese Vorgehensweise wird auch “zero padding” genannt. Die Berechnung der Faltung erfolgt wie gezeigt mit einer endlichen Summe und das Ergebnis könnte wieder im Definitionsbereich eingeschränkt werden. Der Definitionsbereich nach der Faltung ist größer als der ursprüngliche, da sich der Träger ausgedehnt hat. Da wechselnde Intervall-Größen die Rechnung erschweren, könnte man den Bereich nach der Rechnung wieder beschneiden, wobei auch Werte ungleich Null weglassen werden müssten.

Wenn die Werte außerhalb des Definitionsbereiches unbekannt sind, ist es nur eine Näherung, diese auf Null zu setzen. Wenn auch “zero padding” nur eine Näherung darstellt, ist es genauso gut möglich, eine andere Interpretation zu wählen: Wir betrachten die Funktionen als periodisch mit der Größe X des Definitionsbereiches:

$$f(x + X) = f(x)$$

Nun können wir für die endlichen Bildmodelle die Faltung quasi genauso definieren, wie bei den unendlichen. Wir beschränken die Summation beziehungsweise Integration einfach auf den Definitionsbereich:

$$(f * g)_n = \sum_{i=0}^{N-1} f_i g_{n-i} \quad \text{und} \quad (f * g)_{m,n} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_{i,j} g_{m-i,n-j} \tag{2.5}$$

und für analoge Funktionen:

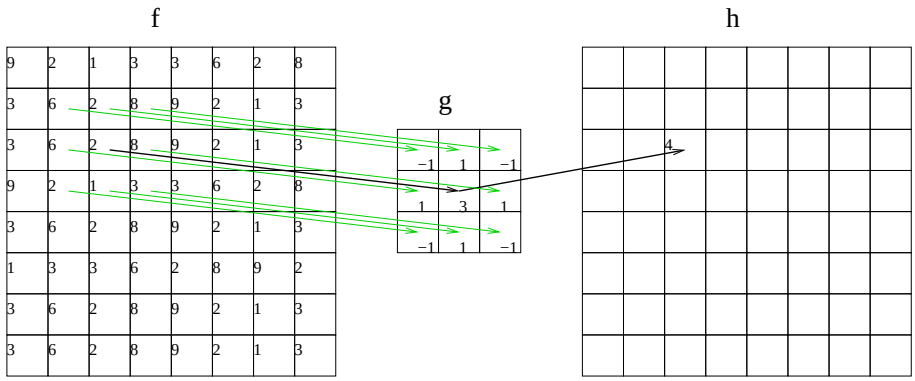
$$(f*g)(t) = \int_0^X f(\xi) g(t-\xi) d\xi \quad \text{und} \quad (f*g)(t_1,t_2) = \int_0^X \int_0^Y f(\xi,\eta) g(t_1-\xi, t_2-\eta) d\xi d\eta \tag{2.6}$$

Wenn hier Werte für g_{n-i} beziehungsweise $g(t-\xi)$ für Argumente außerhalb des Definitionsbereiches $[0, N-1]$ beziehungsweise $[0, X)$ benötigt werden, bestimmen wir sie durch periodische Fortsetzung. Damit haben wir die Faltung auch für endliche Bildmodelle definiert. Diese Definition nennt man auch zyklische Faltung.

2.6. Lineares Filter

In der Bildverarbeitung ist der Begriff des linearen Filters üblich.

Lineares Filter: Gegeben sei ein Bild f und eine Filtermaske g mit fest vorgegebenen Zahlenwerten. Lineare Filterung bedeutet nun, dass wir die Filtermaske g mit ihrem Mittelpunkt über ein Pixel in f setzen und die Produkte der Maskenwerte mit den darunterliegenden Grauwerten von f bilden und summieren. Diese Summe weisen wir dem entsprechenden Pixel im Ergebnisbild h zu. Dies tun wir für alle Pixel des Bildes h. Am Rande setzen wir periodisch fort.



$$h_i = \sum_k f_{i+k} g_k$$

Wir sehen, dass dieses lineare Filter genau die Korrelation beschreibt. Der Unterschied zur Korrelation ist eher technischer Natur: Da man typischerweise von einer Filtermaske ausgeht, die nur eine relativ kleine Größe besitzt, ist es sinnvoll die Summation auf diesen Bereich zu beschränken. Die im mathematischen Modell der Korrelation außerhalb der Maske liegenden Werte von g sind Null und tragen nicht zur Summation bei.

Lineares Filtern ist die Korrelation von f mit g oder die Faltung von f mit der gespiegelten Filtermaske g^R.

Will man nun ein Bild f mit zwei Filtern g und h nacheinander filtern, also $(f \circ g) \circ h$ berechnen, ist es aus Effizienzgründen sinnvoll, die beiden Filter g und h zu einem Filter zusammenzufassen. Leider gilt aber das Assoziativgesetz für die Korrelation nicht. Wir wenden die uns bekannten Rechenregeln an:

$$(f \circ g) \circ h = f * g^R * h^R = f * (g * h)^R = f \circ (g \circ h^R)$$

oder zusammengefasst:

$$(f \circ g) \circ h = f \circ (g \circ h^R) = f \circ (g * h)$$

Das zusammengefasste Filter entsteht durch Korrelation von g mit dem gespiegelten Filter h oder Faltung von g und h.

2.7. Neutrales Element der Faltung

Gibt es eine Funktion n, so dass $f * n = f$?

Dieses neutrale Element ist der Einheitsimpuls δ.

Im diskreten Fall ist dieser einfach zu definieren:

$$\delta_i = \begin{cases} 1 & \text{falls} \quad i = 0 \\ 0 & \text{sonst} \end{cases}$$

Führen wir die Faltung jetzt aus, so sehen wir

$$(f * \delta)_n = \sum_i f_i \delta_{n-i} = f_n$$

Da δ_{n-i} nur dann verschieden von Null ist, wenn $n = i$ ist, verschwinden alle anderen Glieder der Summe.

Für analoge Funktionen ist es schwieriger, das neutrale Element anzugeben. Hier ist δ die Dirac-Funktion. Sie muss die Eigenschaft

$$f(t) = (f * \delta)(t) = (\delta * f)(t) = \int_B f(\tau)\delta(t-\tau)d\tau = \int_B \delta(\xi) f(t-\xi)d\xi \qquad (2.7)$$

erfüllen. Setzen wir für f die spezielle Funktion $f \equiv 1$ ein, erhalten wir die notwendige Bedingung $\int_B \delta(\xi)d\xi = 1$. Die Dirac-Funktion ist fast überall Null, im Ursprung hat sie

jedoch den Wert Unendlich. Integriert man die Funktion muss 1 herauskommen, wenn der Ursprung zum Integrationsbereich gehört:

$$\int_B \delta(x)dx = \begin{cases} 1 & \text{falls} \quad 0 \in B \\ 0 & \text{sonst} \end{cases}$$

Das ist keine "normale" Funktion mehr, man nennt diese Art Funktionen auch verallgemeinerte Funktionen oder **Distributionen**.
Die Dirac-Funktion wird oft als Grenzwert einer Folge von Rechteck-Funktionen veranschaulicht, die immer schmaler und höher werden.

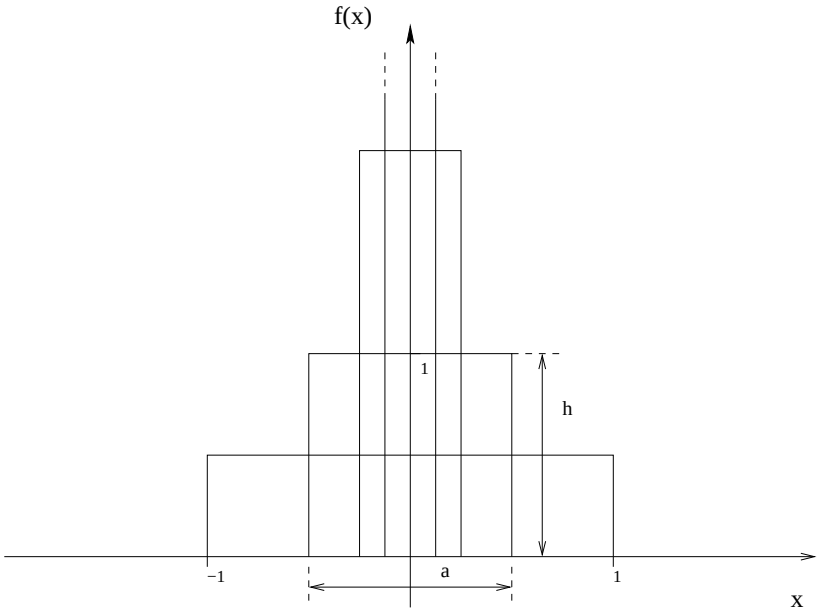


Abbildung 2.1.
Die delta-Funktion als Grenzwert einer Folge von Rechteckfunktionen: $h \cdot a = 1$, $a \to 0$

Die Multiplikation einer Funktion mit der Dirac-Funktion und nachfolgende Integration löst genau einen Funktionswert aus einer Funktion heraus (2.7):

$$f(t) = \int_B f(\tau)\delta(t-\tau)d\tau \tag{2.8}$$

Ist der Einheitsimpuls vielleicht auch neutrales Element der Korrelation? Setzen wir einmal den Einheitsimpuls in die Korrelation ein:

$$f \circ \delta = f * \overline{\delta^R} = f * \delta = f$$

Der Einheitsimpuls ist offensichtlich symmetrisch und rein reell, so dass $\overline{\delta^R} = \delta$ und der Einheitsimpuls ein rechtsseitiges neutrales Element der Korrelation ist.

Auf der linken Seite stehend verhält er sich aber nicht genauso:

$$\delta \circ f = \delta * \overline{f^R} = \overline{f^R}$$

Die Korrelation bewirkt hier also, dass die gespiegelte und konjugiert komplexe Funktion zu f gebildet wird. Ein neutrales Element der Korrelation müsste bei der Faltung also die gespiegelte konjugiert komplexe Funktion bilden. Das geht nicht, ein neutrales Element der Korrelation existiert nicht.

2.8. Inverses Element der Faltung

Das zu einer Funktion g inverse Element g^{-1} soll eine Faltung mit g invertieren, also muss gelten

$$(f * g) * g^{-1} = f * (g * g^{-1}) = f$$

und somit ist $g * g^{-1}$ neutrales Element der Faltung:

$$g * g^{-1} = g^{-1} * g = \delta \tag{2.9}$$

Die Existenz eines inversen Elements ist nicht garantiert.

Haben wir ein System, das eine Funktion f in Form einer Faltung mit einer Funktion g stört ($f' = L_g f = f * g$), so können wir dies korrigieren, falls die inverse Funktion g^{-1} existiert. Dazu berechnen wir die Originalfunktion f als $f = f' * g^{-1}$ aus der verfälschten Funktion f'.

Interessanterweise erlaubt die Kommutativität der Faltung auch noch ein anderes Vorgehen:

$$f = f' * g^{-1} = f * g * g^{-1} = f * g^{-1} * g = f'' * g$$

Wir können auch die Originalfunktion vor dem Durchlaufen des Systems mit $f'' = f * g^{-1}$ vorverzerren, so dass nach dem Durchlaufen des Systems die Originalfunktion vorliegt. Ein typischer Fall für diese Möglichkeit sind analoge Signalübertragungssysteme, die wahlweise vorverzerren oder nachträglich korrigieren.

Bei der Invertierung einer Korrelation ist die Situation etwas anders: Haben wir f mit g korreliert ($f' = f \circ g$), suchen wir eine inverse Funktion h, die bei der Korrelation mit g die Ursprungsfunktion wieder herstellt:

$$f = f' \circ h = (f \circ g) \circ h = f \circ (g * h)$$

Offensichtlich bedeutet das, dass $g * h = \delta$ sein muss. Die gesuchte Funktion h ist also die **Faltungsinverse** von g.

2.9. Interpolation als Faltung

Manchmal wird zur Interpolation einer Funktion eine "gemischte" Faltung angegeben:

$$f^a(x) = (f * g)(x) \;=\; \sum_k f_k g(x - x_k)$$

Die diskrete Funktion f wird mittels g in eine analoge Funktion f^a überführt. Dies ist keine (echte) Faltung, hier werden eine digitale und eine analoge Funktion verknüpft. Damit dies eine Interpolation darstellt, muss die Funktion g eine Reihe von Bedingungen erfüllen. Dazu betrachten wir einmal nur die Stützstellen x_n. Hier muss die berechnete Funktion f^a die gleichen Werte wie f_n besitzen.

$$f^a(x_n) = f_n = \sum_k f_k g(x_{n-k})$$

Jetzt handelt es sich wirklich um eine Faltung zweier digitaler Funktionen. Wir sehen, dass die digitale Funktion $g_k = g(x_k)$ der Einheitsimpuls sein muss. Die analoge Funktion g muss also an der Stelle Null den Wert Eins besitzen und bei allen anderen Stützstellen x_k den Wert Null $x_k = k \cdot \Delta x$.
Geht man zum Beispiel von Stützstellen mit dem Abstand 1 aus ($x_k = k$), so würde die Funktion g_l mit

$$g_l(x) = \begin{cases} x+1 & -1 \le x \le 0 \\ 1-x & 0 \le x \le 1 \\ 0 & sonst \end{cases}$$

eine lineare Interpolation beschreiben (siehe Abb. 2.2).

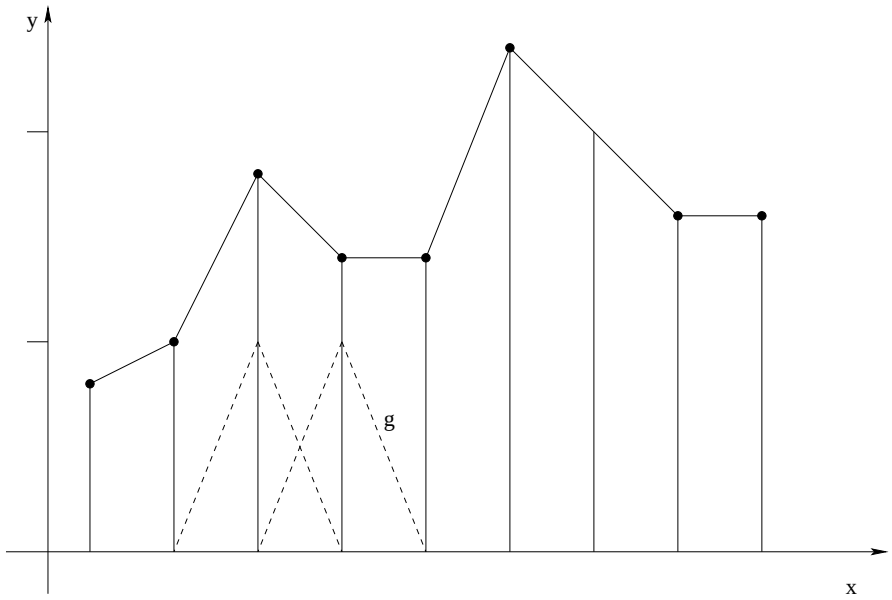


Abbildung 2.2.
Prinzip der linearen Interpolation

Die Stützstellen (Kreise) gehen mit linear abnehmendem Gewicht in die Interpolation ein (gestrichelt dargestellt). Die Gewichte entsprechen der Funktion g, an die Stelle des jeweiligen Wertes verschoben.

2.10. Ableitungen der Faltung

Wir betrachten die Faltung von differenzierbaren Funktionen, also $g(x) = h(x) * f(x)$ und wollen die Regel zur Ableitung von $g(x)$ bestimmen. Es gilt

$$g'(x) = \left(\int_0^X h(\tau) f(x-\tau) d\tau \right)' = \int_0^X h(\tau) f'(x-\tau) d\tau = \int_0^X f(\tau) h'(x-\tau) d\tau$$

Damit gilt die Regel $g'(x) = (h * f)'(x) = h'(x) * f(x) = h(x) * f'(x)$. Die Bestimmung der Ableitung einer durch Faltung entstandenen Funktion kann also dadurch erfolgen, dass einer der Operanden vor der Faltung abgeleitet wird.

Eine praktische Anwendung erfährt diese Regel, wenn ein Bild mit der Gauß-Funktion $G(\mathbf{x})$ als Filter geglättet wird und anschließend auf das erhaltene Bild ein Laplace-Operator Δ angewendet wird. Nach obiger Ableitungsregel gilt dann

$$\Delta \left(G(\mathbf{x}) * f(\mathbf{x}) \right) \;=\; \left(\Delta G(\mathbf{x}) \right) * f(\mathbf{x})$$

Man kann die beiden Schritte also in einem Filter vereinigen, indem die Gauß-Funktion abgeleitet wird.

Mit der isotropen Gauß-Funktion $G(\mathbf{x}) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ bilden wir

$$LoG \;=\; \Delta G(x,y) \;=\; \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Wir erhalten den sogenannten LoG-Filter (Laplace of Gaussian) oder Mexican hat:

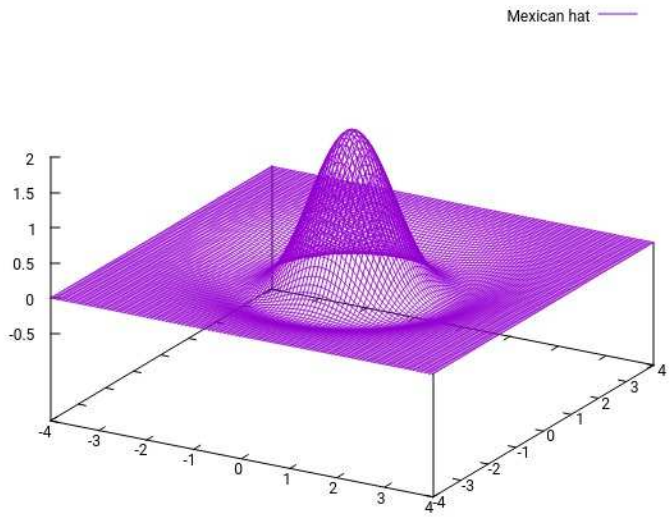


Abbildung 2.3.
Filterfunktion des "Mexican hat"

Für die praktische Anwendungen muss dieses Filter noch diskretisiert werden.

2.11. Wrap-around effect und zero padding

Oft sind Funktionen in einem unendlichen Modell $A1[\infty]$ oder $D1[\infty]$ gegeben, bei praktischer numerischer Berechnung müssen wir aber ein endliches diskretes Modell $D1[N]$ benutzen. Wegen vieler praktischer Vorteile wird hier meist die zyklischen Faltung verwendet. Hier muss man die Fehler durch die periodische Fortsetzung abschätzen. Sind die Originalfunktionen Funktionen mit beschränktem Träger, sind also nur in einem endlichen Teilbereich des Definitionbereichs Funktionswerte verschieden Null, so lassen sich die Funktionen durch eine endliche Zahl von Werten korrekt beschreiben. Hier müsste man doch auch mit den endlichen Modellen korrekt rechnen können?

Betrachten wir noch einmal das Beispiel der Summe zweier Würfelergebnisse aus dem Abschnitt Faltung (2.1): Die Funktion der Würfelwahrscheinlichkeiten besitzt im Intervall $1 \leq i \leq 6$ den Wert $\frac{1}{6}$, alle anderen Werte sind Null. Benutzen wir also zunächst das Intervall $0 \leq i \leq 7$ als (endlichen) Definitionsbereich im Modell D1(8). Wenden wir jetzt die Faltung an, so erhalten wir nicht die erwartete Dreiecksfunktion als Ergebnis.

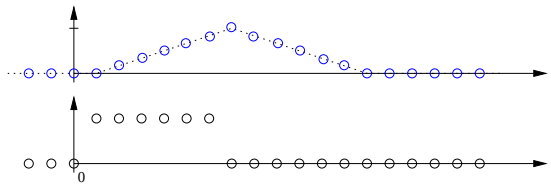


Abbildung 2.4.
Faltung $p * p$ im unendlichen Modell

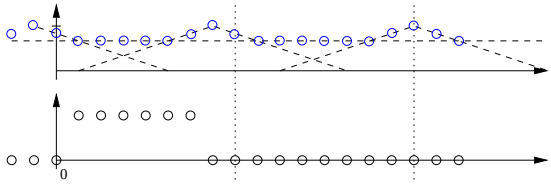


Abbildung 2.5.
Faltung $p * p$ im Modell $D1[8]$

Wie in der Darstellung mittels der gestrichelten Linien angedeutet, überlagern sich die Teile des korrekten Faltungsergebnisses - dies nennt man **wrap around effect**.

Bei Funktionen mit kompaktem Träger lässt sich der wrap around effect aber umgehen. Wir müssen berücksichtigen, dass sich bei der Faltung der Träger ausdehnt, wie wir bereits gezeigt haben (2.5).

Wählen wir das Intervall $[0, 15]$, stimmt die zyklische Faltung im berechneten Intervall mit der unendlichen Faltung im Modell $D1[\infty]$ überein.

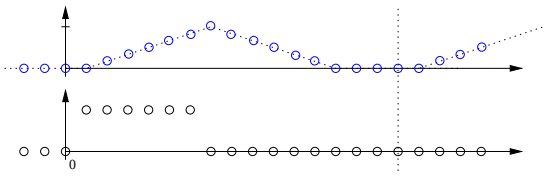


Abbildung 2.6.
Faltung $p * p$ im Modell $D1[16]$

Da die Funktion p nur im Intervall $[1, 6]$ von Null verschiedene Werte besitzt, lassen sich Grenzen für n angeben, so dass nach der Faltung ein von Null verschiedener Wert entsteht (2.5). Es ergibt sich ein Intervall $2 \leq n \leq 12$, was ja schon aus der Aufgabenstellung ersichtlich ist. Schließt der endliche Bereich unseres Modells D1[N] dieses Intervall ein, so ist das Ergebnis der zyklischen Faltung in diesem Bereich gleich der Faltung im unendlichen Modell.

2.12. Schnelle Faltungen im Ortsraum

Wie implementiert man nun Faltungen im Ortsraum? Wir gehen vom Modell D2[M,N] aus. Implementiert man direkt die Definition 2.1, so ergibt sich eine Komplexität von $O(M^2N^2)$.

$$(h * f)_{x,y} = \sum_{i=0}^{M-1} \sum_{k=0}^{N-1} h_{i,k} \cdot f_{x-i,y-k}$$

Im Abschnitt zur linearen Filterung hatte wir bereits gesehen, dass man die Summation auf einen deutlich kleineren Teil beschränken kann, wenn sich die Filtermaske h auf einen

Bereich von $m \cdot n$ beschränkt, anders gesagt, die Funktion nur in einem Bereich dieser Größe von Null verschiedene Werte hat:

$$(h * f)_{x,y} = \sum_{i=0}^{m-1} \sum_{k=0}^{n-1} h_{i,k} \cdot f_{m-i,n-k}$$

Die Komplexität ergibt nun $O(MNmn)$. Geht man dvon aus, dass m und n typischerweise viel kleiner als M und N sind, so entsteht ein beachtlicher Zeitgewinn. Für kleine m und n lassen sich Filterungen so sinnvoll implementieren.

Nehmen wir aber mal an, die Maske h ließe sich schreiben als $h_{i,k} = hx_i \cdot hy_j$, so ergäbe sich für die Faltung

$$g_{x,y} = (h * f)_{x,y} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} hx_i hy_j f_{x-i,y-j} = \sum_{i=0}^{m-1} hx_i \left(\sum_{j=0}^{n-1} hy_j f_{x-i,y-j} \right)$$

Die Faltung $h * f$ würde sich damit auflösen in zwei hintereinander ausgeführte Faltungen mit Masken der Größen $m \cdot 1$ und $1 \cdot n$, aus einer Komlexität $O(MNmn)$ würde $O(MNm + MNn)$.

Filter der Form $h_{i,k} = hx_i \cdot hy_j$ heißen *separierbar*, weil die Faltung dann in zwei "kleine" Faltungen aufspaltbar ist.

- Zwei Beispiele für separierbare Filtermasken sind der Mittelwertfilter und der Gaußfilter:

$$h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

$$h = \begin{pmatrix} 0.11 & 0.32 & 0.11 \\ 0.32 & 1 & 0.32 \\ 0.11 & 0.32 & 0.11 \end{pmatrix} = \begin{pmatrix} 1 \\ 2.95 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0.11 & 0.32 & 0.11 \end{pmatrix}$$

  Auf Grund der speziellen Struktur des Mittelwertfilters, kann dieser noch effizienter implementiert werden.

- Zur Rauschunterdrückung werden oft Binomialfilter verwendet. Binomialfilter approximieren Gaußfilter im diskreten Gitter, und stellen somit eine optimale Quantisierungsstrategie von Gaußfiltern dar. Bei einem eindimensionalen Filter der Ausdehnung $N + 1$ laufen die Pixelpositionen von 0 bis N, daher ist

$$P(X = k) = p_k = \binom{N}{k} p^k q^{N-k}$$

  Wir setzen $p = 1/2$ und erzeugen die zweidimensionalen Filterkoeffizienten durch Multiplikation, also $p_{i,j} = p_i \cdot p_j$. Damit ist die Filtermaske schon von der Konstruk-

tion her separierbar. Es gilt z.B. für einen $5x5$-Binomialfilter:

$$h = \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix} .$$

Wie können wir nun für andere Filter ermitteln, ob diese separierbar sind? Dazu benutzen wir den Approximationssatz bezüglich der Singulär-Wert-Zerlegung (SVD) einer Matrix:

$$B = \sum_{j=1}^{r} \sigma_j u_j \cdot v_j^T$$

Wenn diese Summe nur genau einen Summanden besitzt, dann ist B separierbar, ansonsten ist B die gewichtete Summe von separierbaren Filtermasken. Wenn wir den Approximationssatz beachten, dann können wir also die Filtermaske durch eine geringe Anzahl separierbarer Filtermasken approximieren. Dazu einige Beispiele bekannter Filtermasken und deren singuläre Werte.

- Wir betrachten den $3x3$ Laplace-Filter:

$$h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} , \Sigma = \begin{pmatrix} 8.196 & 0 & 0 \\ 0 & 1.196 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

  Wir sehen der Rang ist zwei, wir hätten also zwei separable Filtermasken.

- Als letztes Beispiel einer Filtermaske betrachten wir den "Mexican Hat":

$$h = \begin{pmatrix} 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & -1 & -3 & -3 & -3 & -1 & 0 \\ -1 & -3 & 0 & 7 & 0 & -3 & -1 \\ -1 & -3 & 7 & 24 & 7 & -3 & -1 \\ -1 & -3 & 0 & 7 & 0 & -3 & -1 \\ 0 & -1 & -3 & -3 & -3 & -1 & 0 \\ 0 & 0 & -1 & -1 & -1 & 0 & 0 \end{pmatrix} , \Sigma = \begin{pmatrix} 28.896 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.722 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.141 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.314 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  Wir sehen, der Rang ist vier. Die beiden letzten Singulärwerte sind aber im Verhältnis zu den beiden ersten sehr klein, so dass wir eine gute Approximation erzielen, wenn wir nur mit den beiden ersten Basisbildern arbeiten.

3. LSI-Operatoren

3.1. Definition der LSI-Eigenschaft

Im folgenden wollen wir LSI-Operatoren (**L**inear **S**hift **I**nvariant) untersuchen. LSI-Operatoren sind **lineare** Operatoren und sie sind **verschiebungsinvariant**.

3.1.1. Linearität

Ein Operator O ist linear, wenn

$$O(\lambda \cdot f + \mu \cdot g) \;=\; \lambda \cdot Of + \mu \cdot Og \qquad \forall \lambda, \mu \in C$$

für beliebige Funktionen f und g gilt.

Es sei darauf hingewiesen, dass lineare Funktionen im Allgemeinen nicht linear in diesem Sinne sind. So ist $f(x) = ax + b$ eine lineare Funktion, aber kein linearer Operator.

Wenden wir einen linearen Operator auf das Nullelement an, so sehen wir, dass dass Ergebnis immer das Nullelement sein muss.

$$O\mathbf{0} = O(\mathbf{0} + \mathbf{0}) = O\mathbf{0} + O\mathbf{0} = 0$$

3.1.2. Verschiebungsinvarianz

Verschiebungsinvaranz bedeutet hier, dass der Operator an jeder Stelle "gleich" wirkt: Verschiebt man die Funktion, verschiebt sich auch (nur) das Resultat.

$$O(S_m f) = S_m(Of)$$

Wir dürfen den LSI-Operator O mit dem Verschiebungsoperator S_m vertauschen, die Reihenfolge spielt keine Rolle.

Bei der Autokorrelation 2.4 hatten wir eine andere Art von Verschiebungsinvarianz kennengelernt: Die Autokorrelations-Funktion ändert sich nicht, wenn die Ausgangs-Funktion verschoben wird.

3.2. Der Faltungsoperator und LSI-Operatoren

Satz: Ein beliebiger Operator O ist genau dann ein LSI-Operator, wenn er ein Faltungsoperator ist.

Wir spalten den Beweis in zwei Teile auf: Jeder Faltungsoperator ist ein LSI-Operator und jeder LSI-Operator lässt sich durch einen Faltungsoperator beschreiben. Im Beweis werden wir uns auf diskrete Funktionen beschränken.

Ein Faltungsoperator ist ein LSI-Operator.
Wie schon gezeigt, ist die Faltung linear:

$$L_h(\lambda_1 f + \lambda_2 g) = (\lambda_1 f + \lambda_2 g) * h = \lambda_1 f * h + \lambda_2 g * h = \lambda_1 L_h f + \lambda_2 L_h g$$

Bezüglich der Verschiebung gilt

$$(L_f S_k g)_n = (f * S_k g)_n = \sum_m f_m (S_k g)_{n-m} = \sum_m f_m g_{n-k-m}$$

$$= (f * g)_{n-k} = (S_k (f * g))_n = (S_k L_f g)_n$$

Also gilt mit

$$L_f S_k g = S_k L_f g$$

die Verschiebungsinvarianz der Faltung.

Ein beliebiger LSI-Operator O ist ein Faltungsoperator.
Eine beliebige diskrete Funktion g mit N Stützstellen entwickeln wir nach einer Basis. Eine trivale Basis besteht aus verschobenen Einheitsimpulsen $e^{(k)} = S_k \delta$. Wir schreiben

$$g = \sum_{k=0}^{N-1} g_k \cdot e^{(k)}$$

Wenden wir den LSI-Operator O auf g an, erhalten wir aufgrund der LSI-Eigenschaft des Operators

$$(Og)_n = \sum_k g_k (O S_k \delta)_n = \sum_k g_k (S_k O \delta)_n = \sum_k g_k (O\delta)_{n-k} = (g * O\delta)_n$$

Dies beschreibt den Operator als Faltung und wir lesen ab:

$$Og = O\delta * g$$

Jeder LSI-Operator O läßt sich also darstellen als Faltung von g mit der Funktion $O\delta$. Diese Funktion $O\delta$ wird auch als Impulsantwort - die Antwort des Systems auf den Einheitsimpuls - bezeichnet. Die Bezeichnung als Punktverbreiterungsfunktion resultiert aus der Interpretation von $O\delta$ als Verbreiterung des Punktes δ (PSF - **P**oint **S**pread **F**unction).

Die Punktverbreiterungsfunktion kann durch Anwendung des Operators O auf den Einheitsimpuls δ ermittelt werden. Dies passt zu der Erkenntnis, dass δ das neutrale Element der Faltung ist.

$$L_g \delta = g * \delta = g$$

3.3. Beispiele von LSI-Operatoren

Beispiel 1: Das optische System einer Kamera ist modellierbar als ideale Abbildung der 3d-Punkte des Raumes in die Bildebene und einen nachfolgenden Operator, der das ideale Abbild in das aufgenommene, im Rechner abgespeicherte Bild transformiert.

Da sich Lichtintensitäten addieren, können wir von einer Linearität dieses Operators ausgehen. Verschiebungsinvarianz heißt, die Punktverbreiterungsfunktion ist unabhängig von der Lage im Bild.

Sehen wir uns das Modell der dünnen Linse an (Abbildung 3.1).

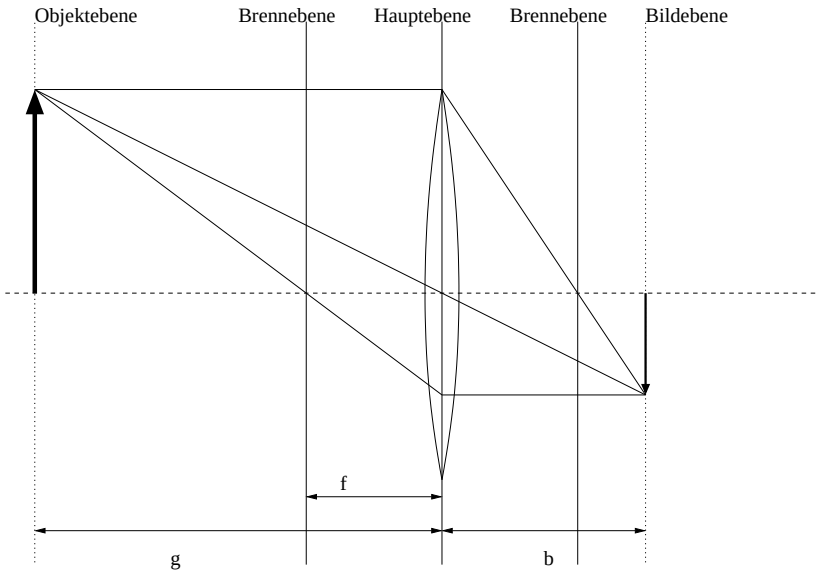


Abbildung 3.1.
Abbildungsmodell der dünnen Linse

Die eingezeichneten Strahlen (achsparalleler Strahl, Mittelpunktstrahl und Brennpunktstrahl) schneiden sich in einem Punkt in der Bildebene, wo ein scharfes Abbild des Objekts entsteht. Zwischen der Brennweite f der Linse und dem Abstand des Objekts (Gegenstandsweite g) und Bildebene (Bildweite b) besteht derZusammenhang

$$\frac{1}{f} = \frac{1}{g} + \frac{1}{b}$$

Die Lage der Bildebene hängt vom Abstand des Objektes ab und Objekte in unterschiedlicher Entfernung sind nicht gleichzeitig scharf.

Befindet sich der Bildsensor nicht in der Ebene des scharfen Bildes eines bestimmten Objekts, wird der Objektpunkt nicht scharf abgebildet, weil die verschiedenen vom Objekt eintreffenden Strahlen an unterschiedlichen Punkten auf den Sensor fallen.

In der Abbildung 3.2 ist die Sensorebene gegenüber der Ebene des scharfen Bildes verschoben. Eingezeichnet sind die extremen Strahlen, die die Linse am Rand passieren. Diese

bilden die Grenze des unscharfen Bildes des Objektpunktes.

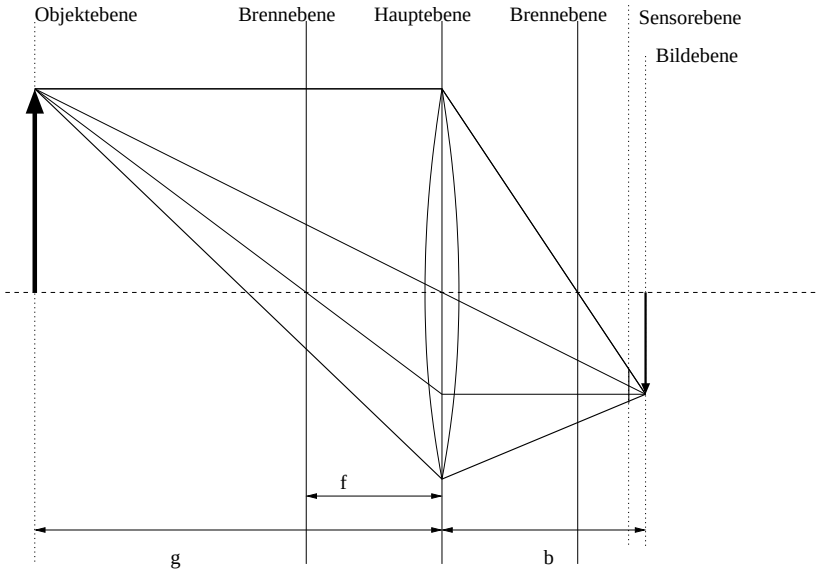


Abbildung 3.2.
Unscharfe Abbildung an der dünnen Linse

Alle Strahlen bilden offensichtlich einen Kegel, dessen Grundfläche die Linse ist, die Spitze liegt in der Ebene des scharfen Bildes. Die Sensorebene schneidet diesen Kegel, das Bild des Objektpunktes ist hier eine Kreisscheibe. Diese Punktverbreiterungsfunktion (PSF) nennt man auch ,,pill box" (Pillendose) . Der Radius hängt offensichtlich von der Verschiebung des Sensorebene zur Bildebene und damit vom Objektabstand ab. Scharf abgebildet werden immer nur Objekte einer Ebene, eine Punktverbreiterungsfunktion ist nur für eine Objektebene gültig. Verschiebungsinvarianz ist deshalb im Allgemeinen nicht gegeben. Finden wir Bereiche, in denen der Abstand der Objektpunkte von der Linse konstant ist, so können wir lokal von der Verschiebungsinvarianz ausgehen und die Unschärfe durch eine Faltung modellieren. Dies ist insbesondere dann gegeben, wenn alle Objekte im Bild sehr weit entfernt sind ("unendlich"), wo sich kleine Entfernungsunterschiede nur gering auswirken.
Dann realisiert das optische System (zumindestens lokal) einen LSI-Operator und

$$g = h * f = L_h f$$

beschreibt die Bildaufnahme hinsichtlich der Unschärfe. Das Originalbild f anhand des aufgenommenen Bildes g und der bekannten PSF $h = L\delta$ zu berechnen, ist Aufgabe der Bildrestauration (Abschnitt 10).

Objektive realer Kameras enthalten mehrere Linsen und weitere Einrichtungen wie Blenden. Dadurch wird das ideale Modell der dünnnen Linse modifiziert. Die Blendenöffnung tritt an die Stelle der kreisförmigen Linse als Basisfläche des Strahlenkegels und ihre Form

bestimmt die Form der Unschärfe [1].

Beispiel 2: Der Verschiebungsoperator S_m ist ein linearer Operator. Um die Verschiebungsinvarianz zu zeigen, müssen wir eine weitere Verschiebung S_k anwenden und zeigen, dass $S_k S_m f = S_m S_k f$ ist. Dies gilt trivialerweise, da sich die Verschiebungen einfach addieren und die Reihenfolge deshalb egal ist. Damit ist der Verschiebungsoperator tatsächlich ein LSI-Operator und wir können ihn als

$$S_m f \;=\; f * S_m \delta$$

darstellen.

Beispiel 3: Der Spiegelungoperator ist ebenfalls linear. Der Test auf Verschiebungsinvarianz liefert:

$$(S_k R f)_n = (Rf)_{n-k} = f_{k-n} = (S_{-k} f)_{-n} = (R S_{-\mathbf{k}} f)_n$$

Der Spiegelungsoperator ist nicht verschiebungsinvariant.

[1] In der Fotografie wird die Form der Unschärfe auch als Bokeh bezeichnet.

4. Zirkularmatrizen

Wir wollen die Faltung im Modell D1(N) jetzt noch einmal mit Mitteln der linearen Algebra formulieren. Endliche diskrete Funktionen mit N Stützstellen sind nichts anderes als N-dimensionale Vektoren. Die Faltung als linearer Operator lässt sich deshalb auch als Multiplikation mit einer $N \times N$-Matrix schreiben:

$$g = h * f = L_h f = C_h \cdot f$$

Die Matrix C_h bestimmen wir gemäß der Definition der Faltung $h * f = \sum_k h_k \cdot S_k f$. Wir finden

$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ . \\ . \\ g_{N-1} \end{pmatrix} = \begin{pmatrix} h_0 & h_{N-1} & h_{N-2} & . & . & h_1 \\ h_1 & h_0 & h_{N-1} & . & . & h_2 \\ h_2 & h_1 & h_0 & . & . & h_3 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ h_{N-1} & h_{N-2} & h_{N-3} & . & . & h_0 \end{pmatrix} \cdot \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ . \\ . \\ f_{N-1} \end{pmatrix}$$

In der ersten Spalte der Matrix finden wir die Funktion h, in den folgenden Spalten die jeweils um eins verschobene Funktion h. Diese zyklische Verschiebung in den Spalten und Zeilen gibt der Matrix den Namen Zirkularmatrix (circulant matrices). Für eine Zirkularmatrix Z gilt $z_{i,j} = z_{i+m,j+m}$, $\forall m$. Wir können jeder Funktion h eine Matrix C_h zuordnen und den Faltungsoperator L_h als Multiplikation mit dieser Matrix ausdrücken.

Als neutrales Element der Faltung haben wir den Einheitsimpuls gefunden. Die Übersetzung in eine Zirkularmatrix liefert, wie zu erwarten, das neutrale Element der Matrixmultiplikation, die Einheitmatrix $C_\delta = E$.

Spiegeln wir eine Funktion, dann wird die Zirkularmatrix transponiert, folglich $C_{f^R} = C_f^T$.

Wenn die inverse Funktion f^{-1} bezüglich der Faltung existiert, gilt $f * f^{-1} = \delta$. Übersetzt in die Matrixdarstellung bedeutet das $C_f \cdot C_{f^{-1}} = E$. Die Zirkularmatrix der inversen Funktion ist also die inverse der Zirkularmatrix der Ausgangsfunktion:

$$C_{f^{-1}} = C_f^{-1}$$

Da die Faltung kommutativ ist $f * g = g * f$, muss das Produkt zweier Zirkularmatrizen ebenfalls kommutativ sein $C_f \cdot C_g = C_g \cdot C_f$ und das Ergebnis wieder eine Zirkularmatrix.

Für zweidimensionale diskrete Funktionen (Bilder) lassen sich genauso die Zirkularmatrizen bilden. Dazu nehmen wir alle Werte des Bildes zeilenweise und tragen sie in einen Spalten-

vektor ein. Hat das Bild die Dimension M, N, dann hat der Vektor die Länge $L = M \cdot N$. Die daraus gebildete Zirkularmatrix hat die Dimension (L, L).

Mit den Zirkularmatrizen haben wir eine erste numerische Möglichkeit, Faltungsgleichungen zu lösen oder überhaupt die Faltung zu berechnen. Auf Grund der hohen Dimension der Zirkularmatrizen hat dies praktisch keine Bedeutung. Dennoch sind die Zirkularmatrizen theoretisch sehr hilfreich, da wir damit Wissen aus der Matrixalgebra auf die Faltung einfach übertragen können.

Dazu ein Beispiel: Die inverse Faltung hat große praktische Bedeutung. Für die Faltungsgleichung $h * f = g$ sind dabei g und h gegeben und die Funktion f ist gesucht. Formuliert mit Zirkularmatrizen sehen wir, dass es sich um die Lösung eines linearen Gleichungssystems handelt:

$$h * f = C_h \cdot f = g$$

Arbeitet man mit realen Werten werden diese oft gestört sein (Rauschen) und die Gleichung hat eventuell gar keine Lösung. Wir wollen sie deshalb im Sinne der kleinsten Quadrate lösen: Wir suchen ein f, so dass $\|h * f - g\|^2 \rightarrow Minimum$ gilt. Mit Zirkularmatrizen formulieren wir $\|C_h \cdot f \ - g\|^2 \rightarrow Minimum$.

Dies ist ein Standardproblem der linearen Algebra und führt auf die *Gaußschen Normalengleichungen*

$$A \cdot x = b \ \rightarrow \quad A^T \cdot A \cdot x = A^T \cdot b$$

Linke und rechte Seite werden mit der transponierten Koeffizientenmatrix A^T multipliziert. Dies tun wir nun auch mit der Zirkularmatrix

$$C_h \cdot f = g \ \rightarrow \quad C_h^T \cdot C_h \cdot f = C_h^T \cdot g$$

Und nun überführen wir diese Gleichungen wieder in Faltungsgleichungen:

$$h * f = g \ \rightarrow \quad h^R * h * f = h^R * g$$

Die lineare Algebra hat uns also geholfen, unsere Faltungsgleichung in eine solche Faltungsgleichung zu überführen, die garantiert eine Lösung besitzt.

Teil III.

Fouriertransformation

5. Mathematische Grundlagen - Basissysteme

In diesem Abschnitt sollen einige wichtige Grundlagen aufgelistet werden, die im weiteren Verlauf verwendet werden. Die Darstellung ist kompakt und nur zur Auffrischung geeignet, weitergehende Informationen liefern gute Mathematikbücher.

5.1. Skalarprodukträume

Wir betrachten einen linearen Raum H mit einem Skalarprodukt.
Ein reellwertiges Skalarprodukt $\langle a, b\rangle \in \mathbb{R}, a, b \in H$ erfüllt folgende Bedingungen:

- $\langle a, a\rangle \geq 0$
- $\langle a, a\rangle = 0 \leftrightarrow a = \mathbf{0}$
- $\langle a, b\rangle = \langle b, a\rangle$
- $\langle \alpha \cdot a + \beta \cdot b, c\rangle = \alpha \langle a, c\rangle + \beta \langle b, c\rangle \quad \text{mit} \quad \alpha, \beta \in \mathbb{R}$
  $\langle a, \alpha \cdot b + \beta \cdot c\rangle = \alpha \langle a, b\rangle + \beta \langle a, c\rangle \quad \text{mit} \quad \alpha, \beta \in \mathbb{R}$

Ein komplexwertiges Skalarprodukt $\langle a, b\rangle \in \mathbb{C}, a, b \in H$ erfüllt folgende Bedingungen:

- $\langle a, a\rangle \geq 0$
- $\langle a, a\rangle = 0 \leftrightarrow a = \mathbf{0}$
- $\langle a, b\rangle = \overline{\langle b, a\rangle}$
- $\langle \alpha \cdot a + \beta \cdot b, c\rangle = \alpha \langle a, c\rangle + \beta \langle b, c\rangle \quad \text{mit} \quad \alpha, \beta \in \mathbb{C}$
  $\langle a, \alpha \cdot b + \beta \cdot c\rangle = \overline{\alpha} \langle a, b\rangle + \overline{\beta} \langle a, c\rangle \quad \text{mit} \quad \alpha, \beta \in \mathbb{C}$

Bekannte Beispiele für Räume mit Skalarprodukt sind:

- Der Vektorraum aller N-dimensionalen Vektoren oder diskreten Funktionen mit N-Stützstellen mit dem Skalarprodukt

$$\langle a, b\rangle = \sum_{i=0}^{N-1} a_i \bar{b}_i$$

- Der Raum aller über B quadratisch integrierbaren, analogen Funktionen mit dem Skalarprodukt

$$\langle a, b\rangle = \int_B a(t)\overline{b(t)}dt$$

Zwei Vektoren $a, b \in H$ sind orthogonal, wenn $\langle a, b\rangle = 0$ ist.
Durch das Skalarprodukt wird eine wichtige Norm definiert:

$$\|a\|^2 = \langle a, a\rangle$$

Wenn $\langle a, b\rangle$ ein Skalarprodukt ist, dann ist auch $\langle a, b\rangle_2 := c \cdot \langle a, b\rangle$ mit $c > 0$ und $c \in \mathbb{R}$ ein Skalarprodukt. Die Anwendung eines solchen Faktors ändert natürlich auch die induzierte Norm. Oft bestimmt man den Faktor so, dass die Norm sinnvoll erscheint.

5.2. Basissysteme

Eine Menge $\{\varphi^k, k \in \mathbb{Z}\}$ von Elementen des Vektorraumes ist eine Basis, wenn die φ^k linear unabhängig sind und durch sie der ganze Raum H aufgespannt wird. Das bedeutet, dass sich jedes Element der Raumes als Linearkombination der Basiselemente $\sum_k \alpha_k \varphi_k$ darstellen lässt. Die Koeffizienten α_k sind eindeutig bestimmt, da die φ_k linear unabhängig sind.
Orthonormalen Basen spielen eine zentrale Rolle. Hier lassen sich die Koeffizienten α_k zur Darstellung eines Elementes des Raumes leicht berechnen.
Eine Basis heißt orthonormal, wenn

$$\left\langle \varphi^k, \varphi^l \right\rangle = \delta_{k-l}$$

gilt, wobei δ_n der Einheitsimpuls (die diskrete Delta-Funktion) ist:

$$\delta_k = \begin{cases} 1, & k = 0 \\ 0, & sonst \end{cases}$$

In orthonormalen Basen sind folglich die φ^k senkrecht zueinander und ihre Norm ist Eins.
Wir entwickeln ein Element f nach einer endlichen orthonormalen Basis φ_k

$$f = \sum_{k=1}^{N-1} \alpha_k \varphi^k$$

Die Koeffizienten α_l können wir sofort mit den Rechenregeln des Skalarproduktes angeben:

$$\left\langle f, \varphi^l \right\rangle = \left\langle \sum_{k=0}^{N-1} \alpha_k \varphi^k, \varphi^l \right\rangle = \sum_{k=0}^{N-1} \alpha_k \left\langle \varphi^k, \varphi^l \right\rangle = \sum_{k=0}^{N-1} \alpha_k \delta_{l-k} = \alpha_l$$

Die Koeffizienten $\alpha_l = \left\langle f, \varphi^l \right\rangle$ nennt man verallgemeinerte Fourierkoeffizienten und deren Bestimmung die verallgemeinerte Fouriertransformation.
Bilden wir nun das Skalarprodukt von f mit sich selbst:

$$\|f\|^2 = \langle f, f\rangle = \left\langle \sum_{k=0}^{N-1} \alpha_k \varphi^k, \sum_{l=0}^{N-1} \alpha_l \varphi^l \right\rangle = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha_k \overline{\alpha}_l \delta_{k-l} = \sum_{k=0}^{N-1} |\alpha_k|^2$$

Die Beziehung

$$\|f\|^2 = \sum_k |\alpha_k|^2$$

wird als Parsevalsche Gleichung oder oft auch als *Energiegleichung* bezeichnet.

Wenn wir diskrete Funktionen mit endlichem Definitionsbereich (Modell D1(N)) betrachten, können wir diese Zusammenhänge auch noch anders formulieren: Wir betrachten die Funktionen als N-dimensionale Vektoren. Die Darstellung einer Funktion als Linearkombination der Basisfunktionen wird dann zu einer Multiplikation des Koeffizentenvektors mit einer Matrix der Basisfunktionen.

$$f = \sum_{k=0}^{N-1} \alpha_k \varphi^k$$

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ . \\ . \\ f_{N-1} \end{pmatrix} = \begin{pmatrix} \varphi_0^0 & \varphi_0^1 & \varphi_0^2 & . & . & \varphi_0^{N-1} \\ \varphi_1^0 & \varphi_1^1 & \varphi_1^2 & . & . & \varphi_1^{N-1} \\ \varphi_2^0 & \varphi_2^1 & \varphi_2^2 & . & . & \varphi_2^{N-1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \varphi_{N-1}^0 & \varphi_{N-1}^1 & \varphi_{N-1}^2 & . & . & \varphi_{N-1}^{N-1} \end{pmatrix} \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_{N-1} \end{pmatrix}$$

$$f = \phi \cdot \alpha$$

Die Spalten der Matrix ϕ sind die orthonormalen Basisfunktionen und damit ist ϕ eine orthogonale bzw. unitäre Matrix.

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ . \\ . \\ f_{N-1} \end{pmatrix} = \begin{pmatrix} (\varphi^0) & (\varphi^1) & (\varphi^2) & . & . & (\varphi^{N-1}) \end{pmatrix} \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_{N-1} \end{pmatrix}$$

$$f = \phi \cdot \alpha \quad \leftrightarrow \quad \alpha = \phi^{-1} \cdot f$$

Die inverse einer unitären Matrix ist die transponierte konjugiert komplexe Matrix, so dass gilt

$$f = \phi \cdot \alpha \quad \leftrightarrow \quad \alpha = \overline{\phi}^T \cdot f$$

Im reellen Fall einer orthogonalen Matrix entfällt das konjugiert komplexe ($\phi^{-1} = \phi^T$).

6. Definition der Fouriertransformation

Die im Abschnitt Basissysteme 5.2 eingeführte verallgemeinerte Fouriertransformation benutzt eine beliebige orthonormale Basis. Die Fouriertransformation selbst basiert auf Basissystemen, die aus sinusförmigen Funktionen bestehen. Sie ist benannt nach dem französischen Mathematiker Jean Baptiste Joseph Fourier (1768-1830). In Abhängigkeit vom verwendeten Bildmodell gibt es vier verschiedene Fouriertransformation.

6.1. Die Fourierreihe

Wir betrachten reellwertige analoge Funktionen im Intervall $[0, X]$ entsprechend dem Bildmodell $A1[X]$. Wir wollen sinusförmige Basisfunktionen verwenden und betrachten deshalb die Sinus- und die Kosinus-Funktion. Wir skalieren diese so, dass im Intervall $[0, X]$ eine ganze Anzahl k Perioden liegen. Wir müssen die Periode der Sinus- und Kosinusfunktion entsprechend strecken oder stauchen und erhalten als Basisfunktionen

$$f^k(x) = \sin(2\pi k \frac{x}{X}) \quad \text{mit} \quad k > 0$$

und

$$g^k(x) = \cos(2\pi k \frac{x}{X}) \quad \text{mit} \quad k \geq 0$$

Um ein Orthonormalsystem zu erzeugen, untersuchen wir die Skalarprodukte der Funktionen:

$$\begin{aligned} \langle f^k, f^m \rangle &= \int_0^X \sin(2\pi k \frac{x}{X}) \sin(2\pi m \frac{x}{X}) dx \\ &= \int_0^X \frac{1}{2}(\cos(2\pi(k-m)\frac{x}{X}) - \cos(2\pi(k+m)\frac{x}{X}))dx \\ &= \begin{cases} \frac{1}{2}X & k = m \\ 0 & sonst \end{cases} \end{aligned}$$

Die Sinusfunktionen sind also orthogonal und müssen nur noch durch einen Vorfaktor normiert werden. Berechnen wir ebenso $\langle g^k, g^m \rangle$ und $\langle f^k, g^m \rangle$, so finden wir als Orthonor-

malsystem:

$$\begin{aligned} g^0 &= \frac{1}{\sqrt{X}} \\ g^k &= \frac{\sqrt{2}}{\sqrt{X}} \cos(2\pi k \frac{x}{X}) \quad \text{mit} \quad k > 0 \\ f^k &= \frac{\sqrt{2}}{\sqrt{X}} \sin(2\pi k \frac{x}{X}) \quad \text{mit} \quad k > 0 \end{aligned}$$

Mittels des Skalarprodukts können wir jetzt die Koeffizienten berechnen:

$$\begin{aligned} a_0 &= \frac{1}{\sqrt{X}} \int_0^X f(x)dx \\ a_k &= \frac{\sqrt{2}}{\sqrt{X}} \int_0^X f(x) \cdot \cos(2\pi k \frac{x}{X})dx \quad \text{mit} \quad k > 0 \\ b_k &= \frac{\sqrt{2}}{\sqrt{X}} \int_0^X f(x) \cdot \sin(2\pi k \frac{x}{X})dx \quad \text{mit} \quad k > 0 \end{aligned}$$

Die Funktion beschreiben wir somit durch:

$$\begin{aligned} f(x) &= a_o g^0 + \sum_{k=1}^{\infty} (a_k g^k + b_k f^k) \\ &= \frac{a_0}{\sqrt{x}} + \sum_{k=1}^{\infty} (a_k \cdot \frac{\sqrt{2}}{\sqrt{X}} \cos(2\pi k \frac{x}{X}) + b_k \cdot \frac{\sqrt{2}}{\sqrt{X}} \sin(2\pi k \frac{x}{X})) \\ &= a_0^* + \sum_{k=1}^{\infty} (a_k^* \cdot \cos(2\pi k \frac{x}{X}) + b_k^* \cdot \sin(2\pi k \frac{x}{X})) \end{aligned}$$

Die Formulierung der letzten Zeile soll deutlich machen, dass die konstanten Faktoren, die sich bei der Herleitung über ein Orthonormalsystem ergeben, auch in die Koeffizienten hinein gezogen werden können. Oft werden Fourierreihen so angegeben. Die Bestimmung der Koeffizienten aus der Funktion wird auch oft **Fourier-Analyse** oder Hintransformation genannt, die Berechnung der Funktion aus den Koeffizienten dementsprechend **Fourier-Synthese** oder Rücktransformation.

Leider benötigen wir zwei unterschiedliche Grundfunktionen, die Sinus- und die Kosinusfunktion. Dies wird schnell plausibel: Die Sinusfunktion ist eine ungerade Funktion ($sin(-x) = -sin(x)$), die Kosinus-Funktion ist eine gerade Funktion ($cos(-x) = cos(x)$). Würde man nur eine der beiden Funktionen verwenden, ließen sich deshalb nur gerade oder nur ungerade Funktionen synthetisieren. Das Symmetrieverhalten der Funktionen ist auch die Ursache dafür, dass negative k für die Funktionen keinen Sinn haben.

6.2. Die komplexe Fourierreihe - die analoge Fouriertransformation im endlichen Intervall (AFT)

Wenn wir zu komplexen Zahlen übergehen, finden wir mit der Eulersche Relation einen sehr schönen Ausdruck, der sowohl den Sinus als auch den Kosinus enthält:

$$e^{i\varphi} = cos(\varphi) + i \cdot sin(\varphi)$$

Wir setzen deshalb Basisfunktionen dieser Form ein. Dazu betrachten wir jetzt komplexwertige Funktionen von reellen Argumenten. Eine reellwertige Funktion, die ein Signal/Bild beschreibt, lässt sich durch einen Imaginärteil mit dem Wert 0 zu einer gleichwertigen komplexwertigen Funktion erweitern.
Wir setzen für die Basisfunktionen an:

$$\varphi^k(x) \;=\; \frac{1}{\sqrt{X}} e^{2\pi i k \frac{x}{X}} \quad \text{mit} \quad k \in \mathbb{Z}$$

Unter Verwendung des komplexen Skalarproduktes $\langle f, g \rangle = \int_0^X f(x)\overline{g(x)}\, dx$ überzeugen wir uns von der Orthonormalität der Basisfunktionen:

$$\begin{aligned}
\left\langle \varphi^k, \varphi^m \right\rangle &= \int_0^X \varphi^k(x)\overline{\varphi^m(x)}dx \\
&= \int_0^X \tfrac{1}{\sqrt{X}} e^{2\pi i k \frac{x}{X}} \overline{\tfrac{1}{\sqrt{X}} e^{2\pi i m \frac{x}{X}}}\, dx \\
&= \int_0^X \tfrac{1}{\sqrt{X}} e^{2\pi i k \frac{x}{X}} \tfrac{1}{\sqrt{X}} e^{-2\pi i m \frac{x}{X}}\, dx \\
&= \int_0^X \tfrac{1}{X} e^{2\pi i (k-m) \frac{x}{X}}\, dx \\
&= \left[\tfrac{1}{X} \tfrac{X}{2\pi i (k-m)} e^{\frac{2\pi i (k-m) x}{X}} \right]_0^X \quad \text{für} \quad k \neq m \\
&= \begin{cases} 1 & \text{für} \quad k = m \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Jetzt können wir die komplexe Fourierreihe einfach angeben:

$$\begin{aligned}
f(x) &= \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x}{X}} \\
\alpha_k \quad &= \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx
\end{aligned} \tag{6.1}$$

Diese Fourierreihenentwicklung in komplexer Schreibweise wird Analoge Fouriertransformation (AFT) genannt. Sie transformiert eine analoge Funktion im endlichen Intervall in ein unendliches diskretes Spektrum $\alpha_k, k \in \mathbb{Z}$:

$$f(x) \quad \leftrightarrow \quad (\alpha_{-\infty}, ..., \alpha_{-1}, \alpha_0, \alpha_1, ...\alpha_{+\infty})$$

Vergleicht man die Definition 6.1 mit anderen aus der Literatur, so findet man oft Formulierungen mit anderen Vorfaktoren. Diese ergeben sich aus unterschiedlichen Definitionen des Skalarproduktes. Wie im Abschnitt Basissysteme 5.2 gezeigt, kann man verschiedene Vorfaktoren vor dem Skalarprodukt verwenden.

Skalarprodukt	Basisfunktionen	Fourierreihe
$\langle f,g\rangle = \int_0^X f(x)\overline{g(x)}dx$	$\varphi^{(k)}(x) = \frac{1}{\sqrt{X}} e^{2\pi i k \frac{x}{X}}$	$f(x) = \frac{1}{\sqrt{N}} \sum_{k=-\infty}^{+\infty} \alpha_k e^{2\pi i k \frac{x}{X}}$ $\alpha_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx$
$\langle f,g\rangle = \frac{1}{X} \int_0^X f(x)\overline{g(x)}dx$	$\varphi^{(k)}(x) = e^{2\pi i k \frac{x}{X}}$	$f(x) = \sum_{k=-\infty}^{+\infty} \alpha_k e^{2\pi i k \frac{x}{X}}$ $\alpha_k = \frac{1}{X} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx$
$\langle f,g\rangle = X \int_0^X f(x)\overline{g(x)}dx$	$\varphi^{(k)}(x) = \frac{1}{X} e^{2\pi i k \frac{x}{X}}$	$f(x) = \frac{1}{X} \sum_{k=-\infty}^{+\infty} \alpha_k e^{2\pi i k \frac{x}{X}}$ $\alpha_k = \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx$

Wir werden immer die zuerst gezeigte symmetrische Variante 6.1 benutzen. “Symmetrie” bezieht sich hier nur auf den Vorfaktor, der für Hin- und Rücktransformation gleich ist. Die Hin- und Rücktransformationen selbst sind natürlich unterschiedlich, da sie zwischen verschiedenen Räumen transformieren.

Da wir die komplexe Fouriertransformation sehr oft dazu benutzen, reellwertige Funktionen (Bilder) zu transformieren, wollen wir einmal die komplexe Fouriertransformierte einer reellwertigen Funktion untersuchen.

Wir bilden α_{-k}:

$$\alpha_{-k} = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i(-k)\frac{x}{X}} dx = \frac{1}{\sqrt{X}} \int_0^X f(x) \overline{e^{-2\pi i k \frac{x}{X}}} dx = \overline{\alpha_k}$$

Hier haben wir $e^{-i\varphi} = \overline{e^{i\varphi}}$ und $\overline{f(x)} = f(x)$ ausgenutzt. Für reelle Funktionen f gilt also $\alpha_{-k} = \overline{\alpha_k}$.

Im zweidimensionalen Fall (Modell A2[X,Y]) formulieren wir entsprechend als Basisfunktionen:

$$\varphi^{(k_1,k_2)}(x,y) = e^{2\pi i k_1 \frac{x}{X}} \cdot e^{2\pi i k_2 \frac{y}{Y}} = e^{2\pi i (k_1 \frac{x}{X} + k_2 \frac{y}{Y})}$$

Die Fouriertransformation für das Modell $A2[X,Y]$ ist damit

$$\begin{aligned} f(x,y) &= \frac{1}{\sqrt{XY}} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \alpha_{k_1,k_2} e^{+2\pi i(k_1 \frac{x}{X} + k_2 \frac{y}{Y})} \\ \alpha_{k_1,k_2} &= \frac{1}{\sqrt{XY}} \int_0^X \int_0^Y f(x,y) e^{-2\pi i(k_1 \frac{x}{X} + k_2 \frac{y}{Y})} dy\ dx \end{aligned} \tag{6.2}$$

6.3. Die endliche diskrete Fouriertransformation (DFT)

Gegeben sei eine diskrete Funktion f_k an N Stützstellen ($k = 0, 1, 2, ..., N-1$). Diese Funktion könnte das Ergebnis der Abtastung einer analogen Funktion an äquidistanten Stützstellen $x_k = k \cdot \Delta x$ sein. Die Stützstellen numerieren wir einfach neu durch mit den Indizes $0, 1, 2, ..., N-1$.

Als Kandidaten für diskrete Basis-Funktionen wählen wir:

$$\varphi_n^{(k)} := \varphi^{(k)}(n) = \frac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}}$$

Das Skalarprodukt definieren wir analog zum stetigen Fall als

$$< f, g >= \sum_{l=0}^{N-1} f_l \overline{g_l}$$

Da Funktionen mit N Stützstellen N-dimensionalen Vektoren entsprechen, muss der Raum N-dimensional sein und eine Basis aus N Vektoren besitzen.

In der Tat finden wir, dass mit

$$\begin{aligned} \varphi_n^{k+N} &= \tfrac{1}{\sqrt{N}} e^{2\pi i(k+N)\frac{n}{N}} \\ &= \tfrac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} e^{2\pi i N \frac{n}{N}} \\ &= \tfrac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} e^{2\pi i n} \\ &= \tfrac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} \\ &= \varphi_n^k \end{aligned}$$

unsere Basisfunktionen periodisch in k sind, also genau N echt verschiedene Funktionen existieren.

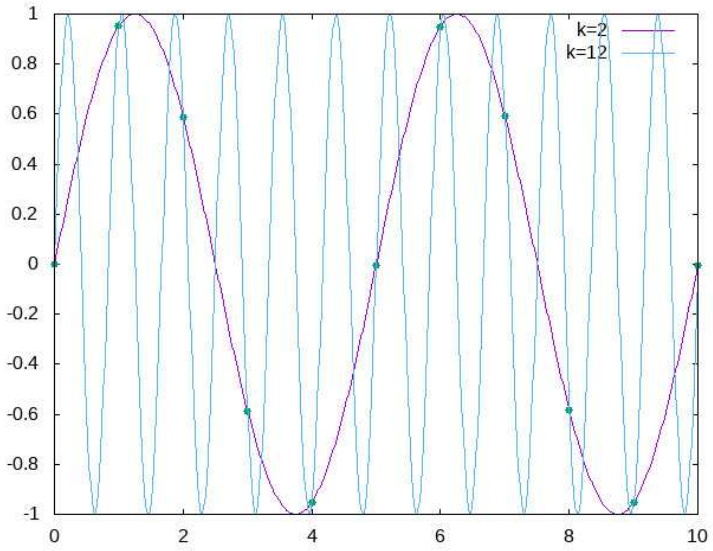


Abbildung 6.1.
Die Funktionen φ_2 und φ_{12} sind im diskreten Definitionsbereich für $N = 10$ identisch.

Wir wählen als Basis die Funktionen φ^0 bis φ^{N-1}, die ein Orthonormalsystem bilden, was wir hier ohne weitere Herleitung akzeptieren wollen.
Daher können wir schreiben

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} \tag{6.3}$$

$$\alpha_k = \left\langle f, \varphi^{(k)} \right\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}} \tag{6.4}$$

6.4. Die Fourier-Integraltransformation (IFT)

Wir gehen hier von der Fouriertransformation für das Bildmodell $A1[X]$ aus. Die Ausgangsfunktion f sei periodisch mit der Periode X. Das ist keine Einschränkung - lassen wir jetzt diese Periode gegen unendlich gehen, ist es keine periodische Funktion mehr. Wegen der Periodizität dürfen wir den Integrationsbereich verschieben:

$$\alpha_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx = \frac{1}{\sqrt{X}} \int_{-\frac{X}{2}}^{+\frac{X}{2}} f(x) e^{-2\pi i k \frac{x}{X}} dx$$

Wenn wir jetzt die Periode gegen unendlich gehen lassen, erhalten wir die Fouriertransformation für den unendlichen Definitionsbereich $[-\infty, +\infty]$.

Dazu substituieren wir $\nu_k = \frac{k}{X}$ und bezeichnen mit $\alpha(\nu_k)$

$$\alpha(\nu_k) \;=\; \sqrt{X} \cdot \alpha_k \;=\; \int\limits_{-\frac{X}{2}}^{+\frac{X}{2}} f(x) e^{-2\pi i \nu_k x} dx$$

Wenn wir nun $\Delta\nu_k := \nu_{k+1} - \nu_k = \frac{1}{X}$ setzen, dann geht die Fourierreihe

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{+2\pi i k \frac{x}{X}}$$

über in

$$f(x) = \sum_{k=-\infty}^{\infty} \alpha(\nu_k) e^{+2\pi i \nu_k x} \cdot \Delta\nu_k$$

Mit $X \to \infty$ geht $\Delta\nu_k \to 0$ und wir erhalten als Grenzwert das uneigentliche Riemann-Integral:

$$f(x) = \lim_{\Delta\nu_k \to 0} \left(\sum_{k=-\infty}^{\infty} \alpha(\nu_k) e^{+2\pi i \nu_k x} \cdot \Delta\nu_k \right) = \int\limits_{-\infty}^{+\infty} \alpha(\nu) e^{+2\pi i \nu x} d\nu$$

Genauso ist dann

$$\alpha(\nu) \;=\; \lim_{X \to \infty} \alpha(\nu_k) \;=\; \int\limits_{-\infty}^{+\infty} f(x) e^{-2\pi i \nu x} dx$$

Damit haben wir die Fourier-Integraltransformation (IFT) hergeleitet:

$$f(x) = \int\limits_{-\infty}^{+\infty} \alpha(\nu) e^{+2\pi i \nu x} d\nu \tag{6.5}$$

$$\alpha(\nu) = \int\limits_{-\infty}^{+\infty} f(x) e^{-2\pi i \nu x} dx \tag{6.6}$$

Durch den Grenzübergang sind die Frequenzen ν jetzt analog! Diese IFT ist symmetrisch in dem Sinne, dass eine analoge Funktion wieder in eine analoge Funktion überführt wird:

$$f(x) \quad \leftrightarrow \quad \alpha(\nu)$$

Man kann also die Fourier-Transformierte $\alpha(\nu)$ selbst wieder transformieren.

6.5. Die unendliche diskrete Fouriertransformation (UDFT)

Wir betrachten nun das Bildmodell $D1[\infty]$, diskrete Funktionen mit dem Definitionsbereich von $-\infty$ bis $+\infty$. Wir wollen diese unendliche diskrete Fouriertransformation aus der Fourierreihe (6.1) herleiten, die wir zunächst noch einmal aufschreiben:

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x}{X}} \\ \alpha_k &= \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx \end{aligned}$$

Das diskrete Spektrum ist eine diskrete Funktion mit unendlichen Definitionsbereich. Wir ersetzen deshalb die α_k mit f_k und nennen die Funktion $f(x)$ nun $\alpha(\nu)$:

$$\alpha(\nu) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} f_k e^{+2\pi i k \frac{\nu}{X}} \quad , \quad f_k = \frac{1}{\sqrt{X}} \int_{-\frac{X}{2}}^{\frac{X}{2}} \alpha(\nu) e^{-2\pi i k \frac{\nu}{X}} d\nu$$

Die Intervalllänge des Spektrums $\alpha(\nu)$ ist nicht vorgegeben. Wir wählen einfach $X = 2\pi$ und erhalten:

$$\alpha(\nu) = \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} f_k e^{+ik\nu} \quad , \quad f_k = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \alpha(\nu) e^{-ik\nu} d\nu$$

Die Transformation hat nicht die Form unseren anderen Transformationen. Wir substituieren deshalb noch einmal $\alpha_d(\omega) := \alpha(-\nu)$ und erhalten die endgültige Form der UDFT:

$$\alpha(\omega) = \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} f_k e^{-ik\omega} \quad , \quad f_k = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \alpha(\omega) e^{+ik\omega} d\omega \tag{6.7}$$

Die UDFT für 2D-Funktionen/Bilder ist:

$$\alpha_d(\omega_1, \omega_2) = \frac{1}{2\pi} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} f_{k_1,k_2} e^{-ik_1\omega_1 - ik_2\omega_2}$$

$$f_{k_1,k_2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \alpha_d(\omega_1, \omega_2) e^{+ik_1\omega_1 + ik_2\omega_2} d\omega_1 d\omega_2$$

7. Eigenschaften der Fouriertransformation

7.1. Bezeichnungen

So wie die betrachtete Ausgangsfunktion bilden auch die Fourierkoeffizienten eine Funktion. Das Argument dieser Funktion ist die Frequenz. Betrachtet man die Funktion der Fourierkoeffizienten, so spricht man auch vom **Frequenzraum** (frequency domain). Im Unterschied dazu vom **Ortsraum** (space domain) oder **Zeitbereich** (time domain), je nachdem ob die Ausgangsfunktionen vom Ort oder der Zeit abhängig sind.

Die Transformation einer Funktion in dem Frequenzraum wird auch als **Hintransformation** bezeichnet, die Gegenrichtung entsprechend als **Rücktransformation**. Die Hintransformation nennt man auch **Fourier-Analyse**, da sie die Funktion in Basisfunktionen unterschiedlicher Frequenz zerlegt. Die **Fourier-Synthese** ist entsprechend die Rücktransformation.

- Die Funktion der Fourierkoeffizienten $\alpha_k(f)$ oder $\alpha_f(\nu)$ nennt man das Fourier-**Frequenzspektrum** oder einfach das **Spektrum** der Funktion f.
- Die Funktion der Beträge der Fourierkoeffizienten $|\alpha_k(f)|$ oder $|\alpha_f(\nu)|$ nennt man das **Amplitudenspektrum** der Funktion f.
- Die Funktion der Quadrate der Beträge der Fourierkoeffizienten $|\alpha_k(f)|^2$ oder $|\alpha_f(\nu)|^2$ nennt man das **Leistungsspektrum** oder Energiespektrum der Funktion f.

Häufig werden bei der Darstellung der komplexen Fourierkoeffizienten die Polarkoordinaten benutzt.

$$\alpha_k = a_k + ib_k = r_k \cdot e^{i\varphi_k}$$

Den Winkel φ nennt man in diesem Kontext auch **Phase**. Die Radien r_k bilden das **Amplitudenspektrum**, die Phasen φ_k das **Phasenspektrum**.

Einen Teilbereich der Frequenzraumes von ν_1 bis ν_2 nennt man **Frequenzband**. Die Differenz $\nu_2 - \nu_1$ ist die **Bandbreite**. Manchmal wird bei Frequenzbändern auch das Frequenzverhältnis in Oktaven angegeben. Ist $0 < \nu_1 \leq \nu_2$, dann drückt O mit $\nu_2 = 2^O \nu_1$ die Bandbreite in Oktaven aus.

Oft werden Bearbeitungen von Funktionen (Filterungen von Signalen) im Frequenzbereich beschrieben. Hier wird im Modell, manchmal auch praktisch, die Funktion in den Frequenzraum transformiert, die Fourierkoeffizienten manipuliert und danach die Funktion zurücktransformiert. Man spricht dabei von **Tiefpassfiltern**, **Bandpassfiltern** und

Hochpassfiltern, je nachdem ob die "tiefen", "mittleren" oder "hohen" Frequenzen "durchgelassen" werden. **Bandsperren** unterdrücken (sperren) ein Frequenzband.

7.2. Nullter Fourierkoeffizient, Gleichanteil

Die nullte Basisfunktion der Fouriertransformation nimmt eine Sonderstellung ein: Sie ist eine konstante Funktion und die einzige, deren Mittelwert nicht Null ist. Hat eine Funktion einen von Null verschiedenen Mittelwert, so wird sich dieser folglich in dem nullten Fourierkoeffizienten ausdrücken.

$$\underset{\alpha_0 \;=\; \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n}{\text{D1[N]}} \qquad \underset{\alpha_0 \;=\; \frac{1}{\sqrt{X}} \int_0^X f(x)dx}{\text{A1[X]}} \qquad \underset{\alpha(0) \;=\; \int_{-\infty}^{+\infty} f(x)dx}{\text{A1[}\infty\text{]}}$$

Die nullten Koeffizienten sind bis auf einen Vorfaktor gleich dem Mittelwert. [1] In der Elektrotechnik wird dieser als *Gleichanteil* von f bzw. als *DC-Komponente* (Direct Current) bezeichnet.
Der Gleichanteil in der Bildverarbeitung ist die mittlere Helligkeit des Bildes. Viele Anwendungen, die Bilder analysieren, benötigen den Gleichanteil nicht. Wird der Gleichanteil weggelassen, ist die Verarbeitung invariant gegen Schwankungen der mittleren Helligkeit, aber nicht des Kontrastes und somit immer noch abhängig von Beleuchtungsschwankungen. Für eine Darstellung von Bildern wird der Gleichanteil natürlich benötigt (es gibt keine negative Helligkeit) und muss dementsprechend erhalten bleiben oder rekonstruiert werden.

7.3. Konvergenzverhalten, Gibbsches Phänomen

Im Falle unendlicher Reihen muss man natürlich das Konvergenzverhalten betrachten. Wir nehmen Modell A1[X] und beschreiben das Konvergenzverhalten der Fourierreihe. Obwohl das Konvergenzverhalten schon lange bekannt war, konnte es erst 1966 von dem Mathematiker Carleson bewiesen werden, wofür er 40 Jahre später, 2006, den Abel-Preis erhielt.

Fourierreihen konvergieren punktweise, wenn sie die Dirichlet-Bedingung erfüllen:

- Das Intervall $[0, X]$ ist in endlich viele Teilintervalle zerlegbar, in denen die Funktion stetig und monoton ist.
- In den (endlich vielen) Unstetigkeitstellen existieren der links- und rechtsseitige Grenzwert $f(x-0)$ und $f(x+0)$.

[1] Der Vorfaktor hängt, wie in Abschnitt 6.2 gesehen, von der Definition des Skalarproduktes ab. Durch geeignete Wahl des Skalarproduktes kann man also auch erreichen, dass der nullte Fourierkoeffizient gleich dem Mittelwert der Funktion ist.

Dann konvergiert die Fourierreihe in jedem Punkt x gegen $f(x)$, wenn die Funktion in x stetig ist und gegen $(f(x-0)+f(x+0))/2$, wenn die Funktion in x in dem Punkt unstetig ist.

Besitzt eine Funktion Sprünge, dann tritt das Gibbssche Phänomen auf. An der Unstetigkeitsstelle treten Über- und Unterschwinger auf, wie in der Abbildung 7.1. Diese verschwinden auch bei steigendem k nicht, ihre Höhe ist unabhängig von k. Es ergibt sich ein relativer Fehler von 18% der Sprunghöhe.

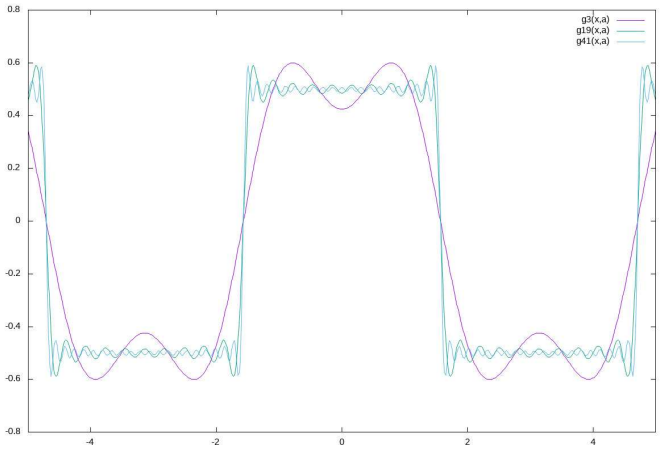


Abbildung 7.1.
Verschiedene Partialsummen der Fourierreihe der Rechteckfunktion

Bei der diskreten Fouriertransformation DFT gibt es kein Konvergenzproblem. Die Summen sind endlich, Unstetigkeiten der Funktionen gibt es nicht. Trotzdem können auch hier Stufen zu Überschwingern führen, wenn die Summation vorzeitig abgebrochen wird. Ein Tiefpass, also die Elimination hoher Frequenzen, ist nichts anderes als ein solcher vorzeitiger Abbruch. Ein "idealer Tiefpass" ist also gar nicht so ideal.

In der Bildverarbeitung hat der Effekt deshalb eine große Bedeutung, da in Bildern häufig Kanten vorkommen, die Sprünge in der Funktion darstellen. Benutzt man jetzt Verfahren, die hohe Frequenzen weglassen, wie zum Beispiel Kompressionsverfahren, beobachtet man das Gibbssche Phänomen (oder **Ringing-**Artefakte) ebenfalls (Abbildung 7.2).

Abbildung 7.2.
Das Originalbild

Abbildung 7.3.
Nach Tiefpassfilterung

7.4. Linearität der Fouriertransformation

Die einfachste Eigenschaft der Fouriertransformation in allen Modellen ist die der Linearität. Es gilt

$$\alpha_k(\lambda f \ + \ \mu g) \ = \ \lambda \cdot \alpha_k(f) \ + \ \mu \cdot \alpha_k(g) \quad \lambda, \mu \in \mathbb{C}$$

7.5. Periodizität

Die Definition der Fouriertransformation im endlichen Intervall benötigt keine Annahme über das Verhalten außerhalb des Definitionsbereichs der Fouriertransformation. Man kann sich jetzt aber fragen, was passiert, wenn man den Definitionsbereich bei der Rücktransformation einfach formal ausdehnt:

$$f(x) = \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x}{X}} \quad \text{für} \quad x \in \mathbb{R}$$

Es ist naheliegend, dass f periodisch mit der Periode X ist, da die Basisfunktionen entsprechend periodisch sind.

$$f(x + mX) = \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x+mX}{X}} = \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x}{X}} e^{2\pi i k m} = \sum_{k=-\infty}^{\infty} \alpha_k e^{2\pi i k \frac{x}{X}} = f(x)$$

Dies kann man analog für die DFT aufschreiben:

$$f_n = \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} \quad \text{für} \quad n \in \mathbb{Z}$$

und

$$f_{n+mN} = \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n+mN}{N}} = \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} e^{2\pi i k m} = \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} = f_n$$

Die Annahme einer periodischen Fortsetzung ist also verträglich mit der Definition der Fourierreihen und der DFT. Wir können die periodische Fortsetzung also ebenso wie bei der Faltung verwenden. Unter dieser Voraussetzung können wir bei der Bestimmung der Fourierkoeffizienten den Integrations- beziehungsweise Summations-Bereich beliebig verschieben, also bei der DFT:

$$\alpha_k = \sum_{n=n_0}^{N+n_0-1} f_n e^{2\pi i k \frac{n}{N}}$$

Im Falle der DFT kann man eine entsprechende Frage für die Fourierkoeffizienten stellen. Wie verhalten sich die Koeffizenten außerhalb des Bereichs?

$$\alpha_k = \sum_{n=0}^{N-1} f_n e^{2\pi i k \frac{n}{N}} \quad \text{mit} \quad k \in \mathbb{Z}$$

Wir haben bereits gesehen, dass sich die Basisfunktionen wiederholen (6.3):

$$\begin{aligned} \varphi_n^{k+N} &= \frac{1}{\sqrt{N}} e^{2\pi i (k+N) \frac{n}{N}} \\ &= \frac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} e^{2\pi i N \frac{n}{N}} \\ &= \frac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} e^{2\pi i n} \\ &= \frac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} \\ &= \varphi_n^k \end{aligned}$$

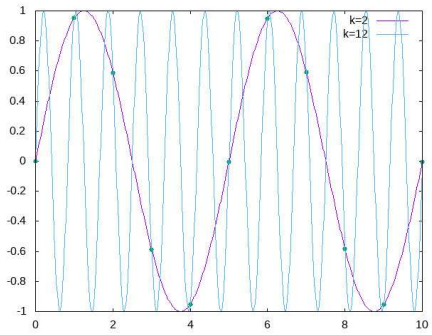


Abbildung 7.4.: Die diskreten Funktionen φ_2 und φ_{12} sind für $N = 10$ identisch.

Damit gilt offensichtlich auch $\alpha_{k+N} = \alpha_k$ (Abbildung 7.4). Die Fourierkoeffizienten sind periodisch. Folglich könnten wir auch hier einen beliebigen Bereich der Länge N für die Frequenz wählen:

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N+k_0-1} \alpha_k e^{2\pi i k \frac{n}{N}}$$
$$\alpha_k = \frac{1}{\sqrt{N}} \sum_{n=n_0}^{N+n_0-1} f_n e^{-2\pi i k \frac{n}{N}}$$

Möchte man mit den Funktionen einfach nur rechnen, so ist das Intervall [0, N-1] eine gute Wahl, beliebige andere Bereiche sind aber auch kein Problem. Wird jedoch ein Bezug zu einer analogen Funktion hergestellt, zum Beispiel weil es sich um eine digitalisierte Funktion handelt, so sind die verschiedenen Frequenzen natürlich nicht gleichwertig. Die Wahl des Intervalls für die Frequenz wird sich hier an der analogen Funktion orientieren. Oft wird hier die betragsmäßig kleinere Frequenz sinnvoller sein, in der Abbildung 7.4 also die zwei statt der 12. Allgemein wäre ein Intervall von [-N/2, +N/2] sinnvoller. Leider ist der Umgang damit umständlicher, da beachtet werden muss, ob N gerade oder ungerade ist.

N gerade	$-N/2+1$ .. $N/2$	$N=4$	-1 .. 2
N ungerade	$-N/2$ .. $N/2$	$N=5$	-2 .. 2

Bei geradzahligen N beobachten wir eine Unsymmetrie des Wertebereichs. Der gespiegelte Wert zu $\alpha_{N/2}$ ist $\alpha_{-N/2} = \alpha_{-N/2+N} = \alpha_{N/2}$, also der Wert selbst. Da für reelle Funktionen gelten muss $\alpha_{-k} = \overline{\alpha_k}$, ist in diesem Fall der $\alpha_{N/2}$ rein reell, der Imaginärteil ist Null.

7.6. Spektrum reeller Funktionen

Ist die zu transformierende Funktion f reell, so haben wir die Eigenschaft $\alpha_k = \overline{\alpha_{-k}}$ bereits hergeleitet. Diese gilt für alle Formen der Fouriertransformation. Dies bedeutet, dass das Amplitudenspektrum einer reellen Funktion eine gerade Funktion ist, also spiegelsymmetrisch zum Koordinatenursprung. Im zweidimensionalen Fall entspricht das einer Drehung von 180° um den Ursprung.
Ist nun die reelle Funktion f auch noch eine gerade Funktion, also ist $f_{-n} = f_n$, so gilt

$$\alpha_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n e^{-2\pi i k \frac{n}{N}} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_{-n} e^{-2\pi i k \frac{n}{N}} = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m \overline{e^{-2\pi i k \frac{m}{N}}} = \overline{\alpha_k}$$

Das Spektrum reeller gerader Funktionen ist also rein reell. Ungerade Funktionen ($f_n = -f_{-n}$) haben ein rein imaginäres Spektrum und es gilt $\alpha_{-k} = -\alpha_k$. Diese Eigenschaften gelten wiederum für alle Fouriertransformationen.

7.7. Parsevalsche Gleichung

Bereits im Abschnitt Basissysteme 5.2 haben wir die Parsevalsche Gleichung kennengelernt, die sich aus der Verwendung eines Orthonormalsystems als Basis ergibt.
Diese Gleichung gilt natürlich auch für alle Arten der Fouriertransformation. Die Fouriertransformation stellt nur einen Wechsel von einem Orthonormalsystem in ein anderes dar, was nichts anderes als eine Rotation ist. Bei Rotationen bleiben die "Längen" der Vektoren erhalten. Wir führen diese Invarianzen nochmals auf:
AFT

$$< f, f > := \int_0^X |f(x)|^2 dx = \sum_{k=-\infty}^{\infty} |\alpha_k|^2 \tag{7.1}$$

DFT

$$< f, f > := \sum_{k=0}^{N-1} |f_k|^2 = \sum_{k=0}^{N-1} |\alpha_k|^2 \tag{7.2}$$

IFT

$$< f, f > := \int_{-\infty}^{\infty} |f(x)|^2 dx = \int_{-\infty}^{\infty} |\alpha(\nu)|^2 d\nu \tag{7.3}$$

Die Parsevalsche Gleichung für die IFT wird oft auch *Satz von Plancherel* genannt.

7.8. Verschiebungstheorem

Wir wenden den Verschiebungsoperator auf eine Funktion f an und betrachten die Auswirkungen auf das Spektrum bei der IFT:

$$\begin{aligned}
\alpha_{(S_{\Delta x} f)}(\nu) &= \int_{-\infty}^{\infty} f(\xi - \Delta x) e^{-2\pi i \nu \xi} d\xi \mid \eta = \xi - \Delta x \\
&= \int_{-\infty}^{\infty} f(\eta) e^{-2\pi i \nu (\eta + \Delta x)} d\eta \\
&= \int_{-\infty}^{\infty} f(\eta) e^{-2\pi i \nu \eta} e^{-2\pi i \nu \Delta x} d\eta \\
&= e^{-2\pi i \nu \Delta x} \int_{-\infty}^{\infty} f(\eta) e^{-2\pi i \nu \eta} d\eta \\
&= \alpha_f(\nu) \cdot e^{-2\pi i \nu \Delta x}
\end{aligned} \tag{7.4}$$

Im diskreten Modell und mit ganzzahligen Verschiebungen erhalten wir entsprechend:

$$\begin{aligned}\alpha_k(S_m f) &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_{n-m} e^{-2\pi i k \frac{n}{N}} \\ &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l e^{-2\pi i k \frac{l+m}{N}} \\ &= \alpha_k(f) e^{-2\pi i k \frac{m}{N}} \\ &= r_k \cdot e^{i\varphi_k} \cdot e^{-2\pi i k \frac{m}{N}} \\ &= r_k \cdot e^{i(\varphi_k - 2\pi k \frac{m}{N})}\end{aligned} \tag{7.5}$$

Die Darstellung in Polarkoordinaten in der letzten Zeile zeigt, dass sich nur die Phase der Fourierkoeffizienten ändert. Das bedeutet aber auch, dass das Amplituden-Spektrum verschiebungsinvariant ist, genauso natürlich das Leistungsspektrum.

7.9. Rotation

In Bildern tritt häufig der Fall einer Rotation ein. Dieses Problem können wir nur im Zweidimensionalen betrachten. Wir benötigen das Modell $A2(\infty, \infty)$ der unendlichen analogen zweidimensionalen Funktionen, da im diskreten Gitter keine echte Rotation existiert. Die in digitalen Bilder auftretenden "Rotationen" sind Approximationen der analogen Rotation. Wir benutzen die IFT 6.5:

$$\alpha_f(\nu_x, \nu_y) = \int_x \int_y f(x,y) e^{2\pi i(\nu_x x + \nu_y y)} dx\ dy \tag{7.6}$$

Für die um φ rotierte Funktion g gilt nun

$$g(x\cos\varphi + y\sin\varphi, -x\sin\varphi + y\cos\varphi) = f(x,y)$$

oder entsprechend

$$g(x,y) = f(x\cos\varphi - y\sin\varphi, x\sin\varphi + y\cos\varphi)$$

Die Fourierkoeffizienten von g sind folglich:

$$\begin{aligned}\alpha_g(\nu_x, \nu_y) &= \int_x \int_y g(x,y)\ e^{-2\pi i(\nu_x x + \nu_y y)} dx\ dy \\ &= \int_x \int_y f(x\cos\varphi - y\sin\varphi, x\sin\varphi + y\cos\varphi)\ e^{-2\pi i(\nu_x x + \nu_y y)} dy\ dx\end{aligned}$$

Wir substituieren:

$$\begin{aligned}u &= x\cos\varphi - y\sin\varphi \\ v &= x\sin\varphi + y\cos\varphi \\ x &= u\cos\varphi + v\sin\varphi \\ y &= -u\sin\varphi + v\cos\varphi\end{aligned}$$

und erhalten:

$$\begin{aligned}
\alpha_g(\nu_x,\nu_y) &= \int_u \int_v f(u,v)\ e^{-2\pi i(\nu_x(u\cos\varphi+v\sin\varphi)+\nu_y(-u\sin\varphi+v\cos\varphi))} dy\ dx \\
&= \int_u \int_v f(u,v)\ e^{-2\pi i(\nu_x u\cos\varphi+\nu_x v\sin\varphi-\nu_y u\sin\varphi+\nu_y v\cos\varphi)} dy\ dx \\
&= \int_u \int_v f(u,v)\ e^{-2\pi i(u(\nu_x\cos\varphi-\nu_y\sin\varphi)+v(\nu_x\sin\varphi+\nu_y\cos\varphi))} dy\ dx \\
&= \alpha_f(\nu_x\cos\varphi-\nu_y\sin\varphi, \nu_x\sin\varphi+\nu_y\cos\varphi)
\end{aligned}$$

Wir sehen: Das Spektrum der rotierten Funktion g ist das rotierte Spektrum der Funktion f.
Es sei darauf verwiesen, dass wir hier nur eine "reine" Rotation um den Ursprung betrachtet haben. Rotationen um andere Punkte stellen Kombinationen der "reinen" Rotation mit Verschiebungen dar.

7.10. Der Spiegelungsoperator und die Fouriertransformation

Wir wollen die Fouriertransformation auf die gespiegelte Funktion anwenden:

$$\begin{aligned}
\alpha(f^R)_n &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k^R e^{-2\pi i k\frac{n}{N}} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_{-k} e^{-2\pi i k\frac{n}{N}} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-2\pi i k\frac{-n}{N}} \\
&= \alpha(f)_{-n} \\
&= \alpha(f)_n^R \\
\alpha(f^R) &= \alpha^R(f)
\end{aligned}$$

Das Spektrum der gespiegelten Funktion ist das gespiegelte Spektrum der Originalfunktion, oder anders formuliert, Fouriertransformation und Spiegelungsoperator können vertauscht werden.

7.11. Die inverse Fouriertransformation der DFT

Die Rücktransformation (Fourier-Synthese) der DFT unterscheidet sich nur geringfügig von der Hintransformation (Fourier-Analyse): einzig das Vorzeichen im Exponenten ist unterschiedlich. Wir wollen einmal sehen, was passiert, wenn wir den Unterschied ignorieren.

Anstatt zurückzutransformieren wenden wir auf das Spektrum noch einmal die Hintransformation an:

$$\alpha_k(\alpha(f)) = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \alpha_m(f) e^{-2\pi i k \frac{m}{N}} = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \alpha_m(f) e^{+2\pi i (-k) \frac{m}{N}} = f_{-k} = f_k^R$$

Das Spektrum des Spektrums ergibt also die gespiegelte Originalfunktion. Offensichtlich gilt auch

$$f = \alpha(\alpha(f))^R = \alpha(\alpha(f)^R) = \alpha(\alpha(f^R))$$

Um mit Hilfe der Hintransformation die Originalfunktion aus dem Spektrum zu errechnen, müssen wir vor oder nach der Transformation spiegeln.
Weiterhin gilt auch

$$\overline{\alpha_l(\overline{\alpha(f)})} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{+2\pi i \frac{lk}{N}} = f_l$$

Für die Rücktransformation kann man das Spektrum also konjugieren, dann die Hintransformation anwenden und erneut konjugieren. Der letzte Schritt kann bei reellwertigen Funktionen f entfallen.
Die gleichen Beziehungen gelten auch für die Integraltransformation. Bei der Fourierreihe und der unendlichen diskreten Fouriertransformation sind diese Beziehungen irrelevant, da Hin- und Rücktransformation zwischen verschiedenen Räumen transformieren und damit nicht austauschbar sind.

7.12. Faltungstheorem

Für Anwendungen in der Bildverarbeitung ist das Faltungstheorem eine der wichtigsten Beziehungen. Es drückt aus, wie sich Faltungen im Ortsbereich im Frequenzbereich auswirken und umgekehrt. Wir wollen dies für die DFT aufschreiben und beweisen.
Faltungstheorem:

$$\alpha_k(f * g) = \sqrt{N} \cdot \alpha_k(f) \cdot \alpha_k(g) \tag{7.7}$$

$$\sqrt{N} \cdot \alpha_k(f \cdot g) = (\alpha(f) * \alpha(g))_k \tag{7.8}$$

Die zweite Zeile ist dual zur ersten Zeile. Wir wollen zunächst zeigen, dass sich die zweite Zeile einfach aus der ersten ergibt. Dazu wenden wir die erste Gleichung auf die Spektren der Funktionen an, ersetzen also f durch $\alpha(f)$ und g durch $\alpha(g)$:

$$\alpha(\alpha(f) * \alpha(g)) = \sqrt{N}\alpha(\alpha(f)) \cdot \alpha(\alpha(g)) = \sqrt{N} \cdot f^R \cdot g^R = \sqrt{N} \cdot (f \cdot g)^R$$

Wir spiegeln beide Seiten:

$$\alpha^R(\alpha(f) * \alpha(g)) = \sqrt{N} \cdot f \cdot g$$

Bilden wir von beiden Seiten das Spektrum, ergibt sich

$$\alpha_k(\alpha^R(\alpha(f) * \alpha(g))) \;=\; \alpha(f) * \alpha(g) \;=\; \sqrt{N} \cdot \alpha(f \cdot g)$$

womit wir die zweite Gleichung aus der ersten hergeleitet haben.
Die erste Gleichung wollen wir jetzt beweisen:

$$\begin{aligned}
\alpha_k(f * g) &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \left(\sum_{m=0}^{N-1} f_m g_{l-m} \right) e^{-2\pi i \frac{kl}{N}} \\
&= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \left(\sum_{m=0}^{N-1} f_m g_{l-m} e^{-2\pi i \frac{kl}{N}} \right) \\
&= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \left(\sum_{m=0}^{N-1} f_m g_{l-m} e^{-2\pi i \frac{k(l-m+m)}{N}} \right) \\
&= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \left(\sum_{m=0}^{N-1} f_m e^{-2\pi i \frac{km}{N}} g_{l-m} e^{-2\pi i \frac{k(l-m)}{N}} \right) \\
&= \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \left(\sum_{l=0}^{N-1} f_m e^{-2\pi i \frac{km}{N}} g_{l-m} e^{-2\pi i \frac{k(l-m)}{N}} \right) \\
&= \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \left(f_m e^{-2\pi i \frac{km}{N}} \sum_{l=0}^{N-1} g_{l-m} e^{-2\pi i \frac{k(l-m)}{N}} \right)
\end{aligned}$$

In die innere Summe geht der Index m mit ein. Wie man aber leicht sieht, bewirkt m nichts als eine Verschiebung, die wegen der Periodizität der e-Funktion und der Funktion g ohne Auswirkung ist und somit entfallen kann. Diese Summe ist also ein konstanter Faktor, den man aus der Summe herausziehen kann und es gilt

$$\alpha_k(f * g) = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m e^{-2\pi i \frac{km}{N}} \cdot \sum_{l=0}^{N-1} g_j e^{-2\pi i \frac{kl}{N}} = \sqrt{N} \alpha_k(f) \cdot \alpha_k(g)$$

Damit haben wir das Faltungstheorem für die DFT bewiesen. Wir überprüfen mit einem einfachen Beispiel auf Plausibilität:

$$\alpha(f * \delta) = \sqrt{N} \cdot \alpha(f) \cdot \alpha(\delta) = \sqrt{N} \cdot \alpha_k(f) \cdot \frac{1}{\sqrt{N}} = \alpha(f)$$

7.13. Korrelation im Frequenzbereich

Da wir die Korrelation auf die Faltung zurückführen können, können wir auch einen entsprechenden "Korrelationssatz" aufstellen:

$$f \circ g = f * \overline{g^R}$$

Da $\alpha_k(\overline{g^R}) = \overline{\alpha_k(g)}$ ist, folgt

$$\alpha_k(f \circ g) \; = \sqrt{N} \cdot \alpha_k(f) \; \cdot \; \alpha_k(\overline{g^R}) = \sqrt{N} \cdot \alpha_k(f) \; \cdot \; \overline{\alpha_k(g)} \tag{7.9}$$

Als Spezialfall können wir das Spektrum der Autokorrelation angeben

$$\alpha_k(f \circ f) \; = \; \sqrt{N} \cdot \alpha_k(f) \; \cdot \; \overline{\alpha_k(f)} \; = \; \sqrt{N} \cdot |\alpha_k(f)|^2 \tag{7.10}$$

Das Spektrum der Autokorrelationsfunktion von f ist das Leistungsspektrum von f. Dieser einfache Zusammenhang wird auch als *Wiener-Chintchin-Theorem* bezeichnet.

7.14. Eigenwerte, Eigenvektoren

Der Faltungssatz zeigt, dass sich Faltungen im Frequenzbereich recht einfach beschreiben lassen. Offensichtlich haben die Basisfunktionen der Fouriertransformation spezielle Eigenschaften bezüglich der Faltung. Wir wenden deshalb einfach mal eine Faltung auf eine Basisfunktion an $f * \varphi^k$. Im Frequenzbereich finden wir:

$$\begin{aligned} \alpha(f * \varphi^k) \; &= \sqrt{N} \cdot \alpha(f) \cdot \alpha(\varphi^k) \\ \alpha(f * \varphi^k)_n &= \sqrt{N} \cdot \alpha(f)_n \cdot \delta_{(n-k)} \\ \alpha(f * \varphi^k)_n &= \sqrt{N} \cdot \alpha(f)_k \cdot \delta_{(n-k)} \end{aligned}$$

Der Ausdruck auf der rechten Seite zeigt, dass nur der k-te Fourierkoeffizient verschieden von Null ist und den Wert $\alpha_k(f)$ hat. Das Gesamtergebnis nach der Rücktransformation ist

$$f * \varphi^k = \sqrt{N} \cdot \alpha_k(f) \cdot \varphi^k$$

Dies hat nicht nur formale Ähnlichkeit mit Eigenwerten und Eigenvektoren: Da wir die Faltung mit f auch als Matrixmultiplikation mit der Zirkularmatrix C_f schreiben können, gilt

$$f * \varphi^k = C_f \varphi^k = \sqrt{N} \cdot \alpha_k(f) \cdot \varphi^k$$

und die Basisfunktionen φ sind Eigenvektoren/Eigenfunktionen jeder Faltung und die Fourierkoeffizienten $\sqrt{(N)} \cdot \alpha(f)$ sind Eigenwerte der Faltung mit f.

7.15. Weiße Funktionen - Whitening

Funktionen werden weiß genannt, wenn ihre Autokorrelation der Einheitsimpuls ist:

$$f \circ f = \delta \tag{7.11}$$

Wir wissen, dass das Spektrum der Autokorrelation das Leistungsspektrum der Funktion ist 7.10. Nun ist $\alpha_k(\delta) = \frac{1}{\sqrt{N}}$. Dass bedeutet, dass bei allen weißen Funktionen die Fourierkoeffizienten des Leistungspektrums wie auch des Amplitudenspektrums konstant sind.

Wenn alle Frequenzen in einen optischen Spektrum gleich stark vertreten sind, spricht man von weißem Licht. Der Name weiße Funktionen wird damit verständlich.

Weiße Funktionen haben eine Reihe interessanter Eigenschaften:

- Die Autokorrelation ist der Einheitsimpuls.
- Das Amplituden- und das Leistungsspektrum sind eine konstante Funktionen.
- $(f \circ f)_n = \langle f, S_n f \rangle = \delta_n$ bedeutet, dass eine weiße Funktion und ihre verschobenen Funktionen eine Orthonormalbasis bilden. Für den Einheitsimpuls hatten wir das bereits kennengelernt und genutzt.
- Die inverse Funktion bezüglich der Faltung zu einer reellen weißen Funktion f ist die gespiegelte Funktion $f^R = Rf$.

$$f \circ f = f * \overline{f^R} = \delta \to \overline{f^R} = f^{-1}$$

Die Idee, eine beliebige Funktion weiß zu machen (whitening), beruht darauf, dass man das Spektrum der Funktion bestimmt und alle Fourierkoeffizienten auf den Betrag $\frac{1}{\sqrt{N}}$ normiert:

$$\beta_k = \frac{\alpha_k}{|\alpha_k|\sqrt{N}} \tag{7.12}$$

Die β_k kann man nun zurücktransformieren. Die weiß gemachte Funktion wird nun der originalen Funktion mehr oder weniger ähnlich sehen.
In praktischen Anwendungen werden einige Fourierkoeffizienten von Funktionen auch den Wert Null besitzen. Wenn wir beim Whitening an einem weißen Bild interessiert sind, welches dem Original möglichst nahe kommt, sollten wir diese Fourierkoeffizienten auf Null setzen und ein "nicht ganz weißes" Ergebnis akzeptieren. Ist unser Ziel dagegen die Konstruktion irgendeiner exakt weißen Funktion, so müssen wir hier einen beliebigen komplexen Wert mit dem Betrag Eins wählen.

7.16. Periodische Strukturen in Bildern

In Bildern mit periodischen Strukturen werden bestimmte Fourierkoeffizienten besonders groß sein, nämlich diejenigen, die gerade die periodischen Verläufe beschreiben. Wenden wir whitening nach 7.12 an, sollten diese Strukturen zurücktreten.
Statt die Amplituden anzugleichen, könnten wir große Beträge auch verstärken, zum Beipiel indem wir alle Fourierkoeffizienten mit ihrem Betrag multiplizieren. Periodische Strukturen im Bild sollten nun besonders hervortreten.

$$\beta_k = \alpha_k \cdot |\alpha_k| \tag{7.13}$$

Diese beiden Filter sind eindeutig nichtlineare Filter. Da die Faktoren vom Bild selbst abhängen, werden solche Filter oft auch als **self-filtering** bezeichnet.

Betrachten wir eine periodische Textur mit Störstellen, dann sollte Whitening die periodische Struktur unterdrücken und die Störungen hervorheben, "self filtering" nach 7.13 die Störungen unterdrücken. Für solch einen Fall ist in den Abbildungen 7.5, 7.6 und 7.7 die Wirkung von self-filtering und whitening zu sehen.

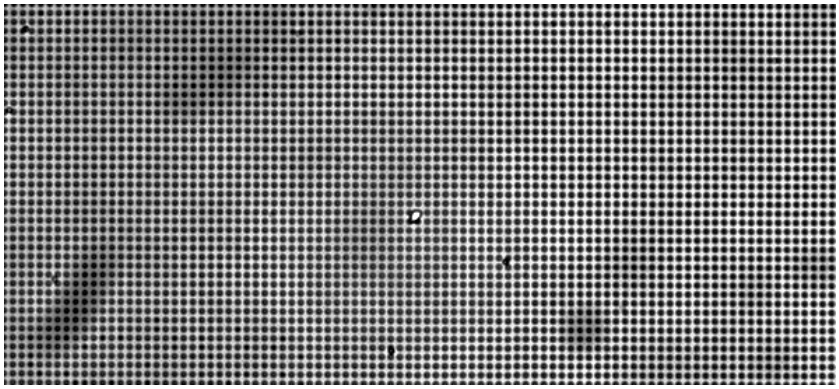


Abbildung 7.5.: Sieb mit Fehlstellen

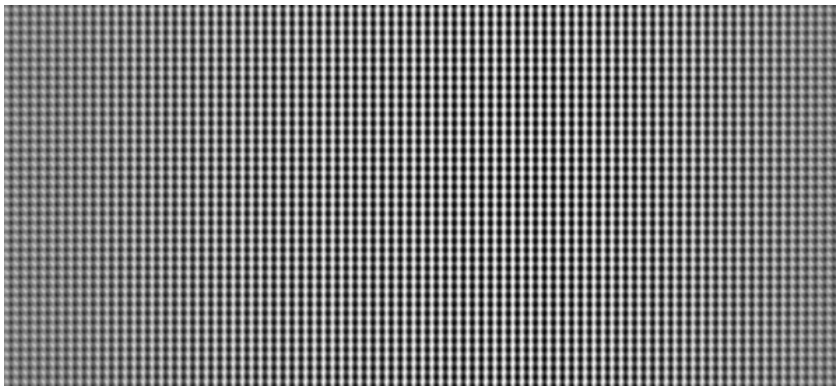


Abbildung 7.6.: Self-Filtering nach 7.13

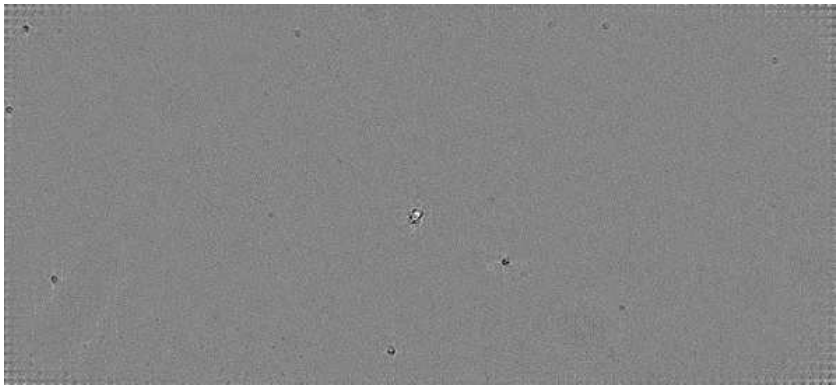


Abbildung 7.7.: Whitening

7.17. Fouriertransformierte spezieller Funktionen

- Kosinusfunktion $f(x) = \cos \omega x, \omega \in \mathbb{R}$
  ω ist eine beliebige reelle Zahl.
  IFT:

$$\begin{aligned}\alpha_{\cos(\omega x)}(\nu) &= \int_{-\infty}^{\infty} \left(\frac{e^{i\omega x} + e^{-i\omega x}}{2} \right) e^{-i\nu x} dx \\ &= \frac{1}{2} \int_{-\infty}^{\infty} e^{ix(\omega-\nu)} dx + \frac{1}{2} \int_{-\infty}^{\infty} e^{-ix(\omega+\nu)} dx \\ &= \frac{1}{2} \left(\delta(\omega - \nu) + \delta(\omega + \nu) \right)\end{aligned}$$

  Für die Sinusfunktion erhält man einen ähnlichen, aber rein imaginären Ausdruck.

  Bei Betrachtung eines endlichen Intervalls $[0, X)$ und Benutzung der Fourier-Reihe (AFT) erhält man ein ähnliches Ergebnis, wenn die Periode des Kosinus ein Bruchteil der Intervalllänge X ist, also $\omega = 2\pi \frac{n}{X}, n \in \mathbb{Z}$. Das Spektrum ist dann unendlich ausgedehnt, aber diskret.

$$\alpha_k(\cos 2\pi \frac{n}{X} x) \;=\; \frac{1}{\sqrt{N}} \cdot (\delta_{k-n} + \delta_{k+n})$$

- Einheitsimpuls

$$\begin{array}{lll} D1(N) & \text{DFT} & \alpha_k(\delta) = \frac{1}{\sqrt{N}} \\ A1(N) & \text{AFT} & \alpha_k(\delta) = \frac{1}{\sqrt{N}} \\ A1(\infty) & \text{IFT} & \alpha_\delta(\nu) = 1 \end{array}$$

  Wir müssen beachten, dass der Einheitsimpuls hier verschiedene Funktionen darstellt. Im Modell D1(N) ist es eine diskrete Funktion, die an der Stelle Null den Wert 1 hat, sonst 0. In den analogen Modellen ist es die Dirac-Funktion. Im Falle der AFT ist diese nur im endlichen Intervall definiert. Wir haben aber schon gesehen, dass eine periodische Fortsetzung der Funktion außerhalb des Intervalls eine sinnvolle Betrachtungsweise ist. Damit wird aus der Dirac-Funktion ein Dirac-Kamm.

- konstante Funktion

$$\alpha(1)_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{-2\pi i k \frac{l}{N}} = \frac{1}{\sqrt{N}} \delta_k$$

- Gauß-Funktion

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \;\leftrightarrow\; \alpha_f(\omega) = e^{-\frac{\sigma^2\omega^2}{2}} \tag{7.14}$$

7.18. Cepstrum

Das Wort Cepstrum wurde aus Spectrum durch Umkehr des "Spec" gebildet. Die Idee ist, das Spektrum einer Funktion zu logarithmieren und danach zurückzutransformieren.

$$c(f) \;:=\; \alpha^{-1}\left[\log \alpha(f)\right]$$

Ohne Logarithmieren würde durch Rücktransformation natürlich die Originalfunktion wieder entstehen. Dieses auch als komplexes Cepstrum bezeichnete Cepstrum hat aber auch ein Problem: Der Imaginärteil des komplexen Logarithmus ist die Phase φ, deren Bestimmung mehrdeutig ist. Das komplexe Cepstrum ist also von der Bestimmung einer sinnvollen Phase (phase unwrapping) abhängig.

$$\log z = \log(r \cdot e^{i\varphi}) = \log r + i\varphi$$

In dieser Form wurde das Cepstrum von Oppenheim 1965 eingeführt [30],[31].
Das power cepstrum wurde bereits 1963 von Bogert, Healy und Tukey [3] eingeführt. Die Verwendung des Leistungsspektrum als Ausgangspunkt umgeht den komplexen Logarithmus:

$$pc(f) := |\alpha^{-1}\left[\log|\alpha(f)|^2\right]|^2$$

Oft wird einfach

$$c(f) := |\alpha^{-1}\left[\log|\alpha(f)|\right]|$$

als [Amplituden-]Cepstrum bezeichnet.
Durch das Logarithmieren werden Produkte in Summen überführt. Dabei werden aber natürlich auch die Faktoren bzw. Summanden verändert. Das Cepstrum wurde ursprünglich benutzt, um in seismologischen, verrauschten und überlagerten Echosignalen die Zeitverzögerung auszurechnen. In der Bildverarbeitung ist es entsprechend möglich, aus zwei überlagerten Bildern die Verschiebung ausrechnen (14.1.7).

8. Numerische Behandlung der Fouriertransformation

8.1. Die schnelle Fouriertransformation - Fast Fourier Transform

Bei einer numerischen Behandlung hat man es mit dem endlichen diskreten Modell zu tun. Die (eindimensionale) diskrete Fouriertransformation (DFT) hatten wir definiert (6.3) als

$$\alpha_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}}$$

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}}$$

Würde man diese Formeln unmittelbar umsetzen, so ergäbe sich für die Hin- wie auch Rücktransformation die Komplexität $O(N^2)$: Es werden N Koeffizienten berechnet und die zu jeweils zu berechnende Summe hat N Glieder. Im zweidimensionalen Fall von Bildern der Größe $M * N$ ergibt sich die Komplexität zu $O(M^2 \cdot N^2)$.
J. W. Cooley und J. W. Tukey veröffentlichten 1965 einen schnelleren Algorithmus [9], der anwendbar ist, wenn sich N in zwei Faktoren aufspalten lässt. Wir wollen die Idee für den Faktor zwei hier betrachten: Ist N eine gerade Zahl, so lassen sich die Werte f_k in Stützstellen mit geradem Index und Stützstellen mit ungeradem Index aufteilen. Wir wollen die resultierenden Teilfunktionen mit g für gerade und u für ungerade bezeichnen (das hat nichts mit geraden und ungeraden Funktionen zu tun).
Wir wollen nun sehen, wie die Fourierkoeffizienten der Funktion f und die Fourierkoeffizienten der Funktionen g und u zusammenhängen. Hat die Funktion f die Dimension N, so haben g und u die Dimension $n = N/2$.

$$\begin{aligned}\sqrt{N}\alpha_k(f) &= \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}} \\ &= \sum_{l=0}^{n-1} f_{2l} e^{-2\pi i k \frac{2l}{2n}} + \sum_{l=0}^{n-1} f_{2l+1} e^{-2\pi i k \frac{2l+1}{2n}} \\ &= \sum_{l=0}^{n-1} g_l e^{-2\pi i k \frac{l}{n}} + e^{-\pi i k \frac{1}{n}} \sum_{l=0}^{n-1} u_l e^{-2\pi i k \frac{l}{n}} \\ &= \alpha_k(g) + e^{-\pi i k \frac{1}{n}} \alpha_k(u)\end{aligned}$$

Die Indizes k der Fourierkoeffizienten von f laufen von 0 bis $N-1$. Da die Funktionen f und u die halbe Dimension besitzen, müssen wir die Periodizität berücksichtigen und führen die Koeffizienten größer gleich n auf das Intervall $[0, n-1]$ zurück:

$$\alpha_k = \begin{cases} \alpha_k(g) + e^{-\pi i k \frac{1}{n}} \alpha_k(u) & \text{für} \quad k < n \\ \alpha_{k-n}(g) + e^{-\pi i k \frac{1}{n}} \alpha_{k-n}(u) & \text{für} \quad k \geq n \end{cases}$$

Da sich α_k und α_{k+n} gut gemeinsam berechnen lassen, da sie die gleichen Fourierkoeffizienten der Funktionen g und u verwenden, formulieren wir:

$$\begin{aligned}\alpha_k &= \alpha_k(g) + e^{-\pi i k \frac{1}{n}} \alpha_k(u) \\ \alpha_{k+n} &= \alpha_k(g) - e^{-\pi i k \frac{1}{n}} \alpha_k(u)\end{aligned} \tag{8.1}$$

Hier laufen die Werte von k von 0 bis $n-1$. Wir haben hier noch verwendet, dass offensichtlich $e^{-\pi i (k+n)\frac{1}{n}} = -e^{-\pi i k \frac{1}{n}}$ ist – der Wert der Phase im Exponenten ändert sich um π, was einer Umkehrung des Vorzeichens entspricht.
Dieses Ergebnis legt die Anwendung eines rekursiven Algorithmus nahe: Die Berechnung der Fourierkoeffizienten einer Funktion erfolgt durch Berechnung der Fourierkoeffizienten der Teilfunktionen mit geradem und ungeradem Index und nachfolgende Vereinigung gemäß 8.1 . Stellt das ursprüngliche N eine Potenz von Zwei dar, lässt sich dir Rekursion auf diese Art bis zur Dimension von 1 fortsetzen. Diese Realisierung des Cooley–Tukey Algorithmus wird auch radix-2 decimation-in-time (DIT) FFT genannt. Die Komplexität dieses Algorithmus ergibt sich aus $log_2 N$ rekursiven Abstiegen und jeweils N Anwendungen von 8.1 zu $NlogN$.
Analoge Formulierungen lassen sich für andere Faktoren finden, wodurch eine Optimierung auch für andere N als Potenzen von zwei möglich sind.
Eine gute Implementierung mit vielen weiteren Optimierung stellt die freie Biliothek fftw3 zur Verfügung.

8.2. Separierbarkeit der mehrdimensionalen FT

Wir betrachten speziell die zweidimensionale DFT:

$$\begin{aligned}
\alpha_{km} &= \frac{1}{\sqrt{X}}\frac{1}{\sqrt{Y}}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} f_{xy}\cdot e^{-2\pi i(k\frac{x}{X}+m\frac{y}{Y})} \\
&= \frac{1}{\sqrt{X}}\frac{1}{\sqrt{Y}}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} f_{xy}\cdot e^{-2\pi ik\frac{x}{X}}\cdot e^{-2\pi im\frac{y}{Y}} \\
&= \frac{1}{\sqrt{X}}\sum_{x=0}^{X-1}\left(\frac{1}{\sqrt{Y}}\sum_{y=0}^{Y-1} f_{xy}\cdot e^{-2\pi im\frac{y}{Y}}\right)e^{-2\pi ik\frac{x}{X}}
\end{aligned}$$

Der Klammerausdruck in der letzten Zeile ist die Fouriertransformierte der Spalte x. Wir können also zunächst alle Spalten transformieren, dann die entstehenden Spalten zeilenweise. Dies vereinfacht die Implementierung, indem nur eine eindimensionale Transformation benötigt wird.
Implementiert man die 2D-Transformation naiv gemäß der ersten Zeile, so benötigt man für die $X \cdot Y$ Fourierkoeffizienten jeweils $X \cdot Y$ Schleifendurchläufe, die Komplexität ist $X^2 \cdot Y^2$. Für die Transformation der X Spalten benötigt man jeweils $Y \cdot Y$ Schleifendurchläufe mit der Komplexität $O(X \cdot Y^2)$. Die nachfolgende Transformation der Zeilen hat dementsprechend die Komplexität $O(Y \cdot X^2)$, insgesamt also $O(X \cdot Y^2 + Y \cdot X^2)$. Damit reduziert sich die Komplexität, was für den Fall $X = Y$ mit $O(X^3)$ statt $O(X^4)$ noch deutlicher wird.
Wendet man hingegen die schnelle Fouriertransformation 8.1 an, so ergibt sich durch die Separierbarkeit kein weiterer Vorteil in der Komplexität: $O(XYlogY + XYlogY) = O(XYlog(XY))$. Der Vorteil einer einfacheren Handhabung durch Verwendung der eindimensionalen Fouriertransformation bleibt aber auch hier.

Teil IV.

Inverse Faltung und Bildrestauration

9. Inverse Faltung

Wenn ein Signal f ein System durchläuft, welches eine Veränderung/Verfälschung des Signals verursacht, möchte man dies korrigieren, um das unverfälschte Signal wiederherzustellen. In LSI-Systemen kann die Verfälschung durch den Faltungsoperator beschrieben werden:

$$g = L_h f = h * f \tag{9.1}$$

Die Funktion h wird im Kontext der Bildverarbeitung meist Punktverbreiterungsfunktion (Point Spread Function oder PSF) genannt, sonst auch Impulsantwort, Gewichtsfunktion oder Stoßantwort. Sind also das verfälschte Signal g und die Punktverbreiterungsfunktion h bekannt, so ist die Suche nach der Originalfunktion f die Umkehroperation zur Faltung, die inverse Faltung oder Entfaltung (deconvolution). Erste Überlegungen zu dieser Problematik hatten wir bereits im Abschnitt zum inversen Element der Faltung (2.8) angestellt. Alternativ zur direkten Lösung der Gleichung 9.1 könnte man auch das inverse Element h^{-1} der Faltung suchen und damit f rekonstruieren.

$$f = h^{-1} * g \quad \text{mit} \quad h^{-1} * h = \delta \tag{9.2}$$

Hier wollen wir uns nun auf praktische Verfahren konzentrieren und betrachten das endliche digitale Funktionsmodell. Wie bisher formulieren wir zur Vereinfachung das meiste eindimensional, auch wenn wir es hauptsächlich auf Bilder, also zweidimensionale Funktionen anwenden wollen.

9.1. Gleichungssystem mit Zirkularmatrizen

Wie bereits diskutiert, können wir die Faltung im diskreten endlichen Modell mittels Zirkularmatrizen darstellen (4).

$$g = L_h f = h * f = C_h f$$

Wenn g und h und damit C_h gegeben sind und f gesucht wird, handelt es sich um ein lineares Gleichungssystem. Damit wäre das Problem grundsätzlich gelöst. Es gibt zahlreiche Verfahren zur Lösung von Gleichungssystemen. Leider ist dies praktisch nicht so einfach: Die Dimension der Vektoren entspricht der Zahl der Werte, in Bildern sind dies oft Millionen. Wir wollen deshalb nach Alternativen mit geringerem Aufwand suchen.

9.2. Iterative Entfaltung

Nehmen wir an, wir kennen eine Näherungslösung f^n für die Funktion f in Gleichung 9.1. Als erste Näherung könnte oft g verwendet werden. Wir können als ein Maß für den Fehler die Differenz $g-h*f^n$ bestimmen. Wir bestimmen jetzt die nächste Näherung f^{n+1}, indem wir diesen Fehler zu f^n addieren:

$$f^{n+1} = f^n + \beta(g - h * f^n)$$

β ist ein Wert, mit dem wir den Verlauf der Iteration beeinflussen. Leicht erkennbar ist: Wenn die Iteration gegen einen Fixpunkt f^* konvergiert, so erfüllt dieses f^* die Gleichung 9.1 und ist das gesuchte f:

$$f^* = f^* + \beta(g - h * f^*) \;\rightarrow\; g - h * f^* = 0 \;\rightarrow\; h * f^* = g$$

Diese Iterationen werden oft als Van Cittert-, Bially- oder Landweber-Iterationen bezeichnet ([42],[20]).
Es existiert ein simples, hinreichendes Kriterium für die Konvergenz der Iteration:
Satz: Die Bedingung

$$\sum_{l=1}^{N-1} |h_l| < |h_0|$$

ist hinreichend dafür, dass h^{-1} bezüglich der Faltung existiert und die Faltungsiteration

$$f^{n+1} = f^n + \frac{1}{h_0}(g - h * f^n)$$

mit einer beliebigen Startfunktion f^0 eine Funktionenfolge $f^{(j)}$ erzeugt, die

$$\lim_{n\to\infty} f^n = f \quad \text{und} \quad g = h * f$$

erfüllt. Bei genauerer Betrachtung stellt diese Bedingung eine sehr starke Einschränkung dar: Eine solche Faltungsmaske überträgt fast das gesamte Pixel an dieselbe Stelle im Zielbild und nur weniger verteilt sich auf alle anderen Pixel.

9.3. Beschränkte Entfaltungskerne

Oft sind die Punktverbreiterungsfunktionen nicht sehr ausgedehnt und Funktionswerte mit größerem Abstand vom Ursprung sind Null. Bei der Lösung der Gleichung $g = h * f$ lässt sich das nicht ausnutzen. Einzig in der Zirkularmatrix C_h sind viele Elemente 0. Günstiger sieht es aus, wenn wir die Faltungsinverse h^{-1} suchen:

$$h * h^{-1} = \delta$$

Zwar ist die Dimensionalität damit immer noch N, aber jetzt ist eine Annahme für h^{-1} möglich: ist h nicht sehr ausgedehnt, so wird auch h^{-1} nicht weit ausgedehnt sein. Das ist

keine exakte Annahme, aber man kann für Werte von h^{-1} fern des Ursprungs den Wert Null ansetzen. Die Lösung wird dann nicht unbedingt exakt sein, aber wahrscheinlich eine gute Näherung liefern. Eine solche Funktion q als Näherung für h^{-1} lässt sich über die Methode der kleinsten Quadrate $\|\delta - h * q\|^2 \rightarrow Minimum$ bestimmen. Nach Aufstellen der Gaußschen Normalengleichungen erhalten wir ein lineares Gleichungssystem von der Dimension der Filtermaske q. Die Lösung können wir als Approximation von h^{-1} benutzen und erhalten als Näherungslösung $\hat{f} = q * g$.

9.4. Mathematische Grundlagen - Pseudoinverse

Wir wollen die Idee der pseudoinversen Matrizen hier informativ einführen. Wir beweisen nicht alle Schritte und beleuchten nicht alle Hintergründe.
Gegeben ist ein lineares Gleichungssystem:

$$y = Ax \tag{9.3}$$

Bei der Lösung können zwei Probleme auftreten:

* Es existiert keine Lösung, weil das Gleichungssystem überbestimmt oder in sich widersprüchlich ist.
  Kann man nicht die Gleichheit erreichen, so versucht man mit der "Methode der kleinsten Quadrate" die Abweichungen davon klein zu halten. Die Forderung

  $$\|Ax - b\|^2 \rightarrow Minimum \tag{9.4}$$

  führt als Lösung zu den Gaußschen Normalengleichungen: Die linke und rechte Seite des Gleichungssystems werden mit A^T multipliziert:

  $$A^T Ax = A^T y \tag{9.5}$$

  Dieses System hat immer eine Lösung und wenn A regulär ist, ist die Lösung der Gaußschen Normalengleichungen 9.5 gleich der Lösung des Gleichungssystems 9.3.

* Es existieren mehrere Lösungen.
  Will man trotzdem eine Lösung angeben, so muss man mit einer Zusatzbedingung eine Lösung auswählen, zum Beispiel die betragskleinste Lösung. Dies ergibt als Gesamtbedingung

  $$x^+ = argmin(\|x\|^2) \quad \text{bei} \quad \|Ax - b\|^2 \rightarrow Minimum \tag{9.6}$$

Die Lösung x^+ nennt man auch Pseudonormallösung von $Ax = y$. Die Matrix A^+, die mit $x^+ = A^+ y$ zur Pseudonormallösung führt, ist die pseudoinverse Matrix.

Als einfaches Beispiel der Pseudoinversen betrachten wir eine $m \times n$ Diagonalmatrix D der Form:

$$D = \begin{pmatrix} a_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & a_{rr} & . & 0 \\ . & . & . & . & . & . & . \\ 0 & . & . & . & . & 0 & 0 \end{pmatrix}$$

Es soll $m > r$ sein. Das mit der Matrix aufgebaute Gleichungssystem

$$y = Dx$$

hat die sehr einfache Form:

$$y_i = \begin{cases} a_{ii}x_i & i \leq r \\ 0 & \text{sonst} \end{cases}$$

Man erkennt sofort: Für $i \leq r$ ist die Lösung bestimmt zu $x_i = \frac{y_i}{a_i i}$. Für die x_i mit $i > r$ gibt es unendlich viele Lösungen: Da y überhaupt nicht davon abhängt, können wir jeden Wert einsetzen. Wenn man gar nichts weiß, ist oft Null die beste Lösung - die betragskleinste. Nun können wir die pseudoinverse Matrix aufschreiben:

$$D^+ = \begin{pmatrix} \frac{1}{a_{11}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{a_{22}} & 0 & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & \frac{1}{a_{rr}} & . & 0 \\ . & . & . & . & . & . & . \\ 0 & . & . & . & . & 0 & 0 \end{pmatrix}$$

Für beliebige Matrizen A benutzt man die Singulärwertzerlegung (SVD). Jede Matrix lässt sich schreiben als

$$A = U\Sigma V^T$$

Dabei sind U und V orthogonale Matrizen und Σ ist eine Diagonalmatrix. Um die [pseudo-]inverse Matrix zu $U\Sigma V^T$ zu bestimmen, müssen wir zunächst die Matrizen einzeln invertieren: Bei orthogonale Matrizen ist dies die transponierte Matrix, für Diagonalmatrizen kennen wir die pseudoinverse. Beim Zusammensetzen kehrt sich die Reihenfolge um:

$$A^+ = V\Sigma^+ U^T$$

9.5. Invers-Filter

Betrachten wir wieder die Aufgabe der Faltungsinvertierung (9.2), so können wir die pseudoinverse Matrix sofort anwenden:

$$\begin{aligned} h * h^{-1} &= \delta \\ C_h h^{-1} &= \delta \\ h^{+} &= C_h^{+} \delta \end{aligned}$$

Die pseudoinverse Funktion h^+ erfüllt

$$h^+ = argmin\|\tilde{h}\|^2 \quad \text{bei} \quad \|\tilde{h} * h \; - \; \delta\|^2 \; \rightarrow \; Minimum$$

und die Pseudonormallösung lässt sich als $f^+ = h^+ * g$ bestimmen.

Die Lösung über Matrizen und Gleichungssysteme hatten wir für Bilder wegen der hohen Dimension als unbrauchbar eingeschätzt. Wir können uns jedoch daran erinnern, dass das Faltungstheorem (7.7) die Behandlung der Faltung im Frequenzbereich sehr vereinfacht: aus der Faltung der Funktionen wird eine Multiplikation der Fourierkoeffizienten. Für die Faltungsinverse gilt demnach:

$$\begin{aligned} h * h^{-1} &= \delta \\ \sqrt{N} \cdot \alpha(h) \cdot \alpha(h^{-1}) &= \alpha(\delta) \\ \sqrt{N} \cdot \alpha(h) \cdot \alpha(h^{-1}) &= \frac{1}{\sqrt{N}} \\ \alpha(h^{-1}) &= \frac{1}{N} \frac{1}{\alpha(h)} \quad \text{mit} \quad \alpha(h) \neq 0 \end{aligned}$$

Was passiert nun aber, falls $\alpha(h) = 0$ ist? Die inverse Funktion h^{-1} existiert dann nicht. Wenn wir aus Erfahrung mit der pseudoinversen Diagonalmatrix in diesem Fall die Null als Wert verwenden, erhalten wir das Invers-Filter:

$$\alpha_k(h^+) = \begin{cases} \frac{1}{N} \frac{1}{\alpha_k(h)} & \alpha_k(h) \neq 0 \\ 0 & \alpha_k(h) = 0 \end{cases} \tag{9.7}$$

Dieses der pseudoinversen Matrix entsprechende Filter wird in diesem Kontext Invers-Filter genannt. *Als Pseudoinvers-Filter wird das Ergebnis der Restauration unter Zwang bezeichnet (Abschnitt 10.2).*
Alternativ können wir das Filter auch direkt auf g anwenden und als Pseudonormallösung $f^+ = h^+ * g$ schreiben:

$$\alpha_k(f^+) = \begin{cases} \frac{1}{\sqrt{N}} \frac{\alpha_k(g)}{\alpha_k(h)} & \alpha_k(h) \neq 0 \\ 0 & \alpha_k(h) = 0 \end{cases} \tag{9.8}$$

10. Bildrestauration

10.1. Das Problem der Bildrestauration

Gegenüber dem idealen Faltungsmodell 9.1 kommt in realen Systemen zur Bilderfassung Rauschen hinzu:

$$g = L_h f + N = h * f + N \tag{10.1}$$

Würde man das konkrete Rauschen kennen, wäre dies kein großes Problem: wir können das Rauschen abziehen und die Faltung invertieren:

$$f^+ = h^+ * (g - n)$$

Leider ist die Welt nicht ideal und das Rauschen ist uns unbekannt. Wenn wir jetzt die inverse Filterung anwenden, erhalten wir

$$\tilde{f} = h^+ * g = h^+ * (h * f + N) = h^+ * h * f + h^+ * N = f^+ + h^+ * N$$

Mit f^+ liefert uns das inverse Filter zwar die im Sinne der Pseudoinvers-Bedingung 9.6 optimale Lösung, der Term mit dem Rauschen $h^+ * N$ könnte aber durchaus dazu führen, dass $\tilde{f}$ eine schlechtere Lösung für f darstellt, als zum Beispiel g.
Man muss also versuchen, das Rauschen in den Algorithmus einzubeziehen. Bildrestaurations-Algorithmen tun dies auf verschiedenste Weise. Dabei hängt viel davon ab, was man überhaupt über das Rauschen weiß und welche Art des Rauschens auftritt.

10.2. Restauration unter Zwang

Wir nehmen als Bildentstehungsmodell

$$g = h * f + N$$

an, wobei die Punktverbreiterungsfunktion h gegeben sein soll. Über das Rauschen wissen wir sehr wenig, wir wollen einzig eine obere Schranke für das Rauschen annehmen $S \geq E(\|N\|^2)$. Wir gehen zur Norm über und suchen ein f, das die Ungleichung

$$\|g - h * f\|^2 \leq S$$

erfüllt. Nun wird es dafür meist unendlich viele Lösungen f geben. Wir suchen aus den vielen möglichen Lösungen eine aus, die unseren Erwartungen entspricht. Das Kriterium dafür, ein Gütemaß, verstecken wir abstrakt in einem Operator T:

$$\|Tf\|^2 \to Minimum \quad \text{bei} \quad \|g - h * f\|^2 \leq S$$

Wir haben also ein Optimierungsproblem zu lösen. Wenn wir den Operator T auf einen linearen Operator L einschränken, dann kann man beweisen, dass das Minimum auf dem Rand angenommen wird. Wir müssen also lösen

$$\|Lf\|^2 \to Minimum \quad \text{bei} \quad \|g - h * f\|^2 = S$$

Da die Restriktion nun in Gleichungsform vorliegt, können wir die Lagrangesche Funktion bilden

$$z(f, \kappa) = \|Lf\|^2 - \kappa(\|g - h * f\|^2 - S) \to Minimum$$

Zur Minimum-Bestimmung müssen nun den Gradienten nach f bilden und gleich Null setzen. Dazu nehmen wir weiter an, dass $L = L_l$ ein LSI-Operator ist. Wir beschreiben die Anwendung von L_l als Multiplikation mit der Zirkularmatrix C_l. Analog sei C_h die Zirkularmatrix zu h. Der Gradient ist somit

$$\nabla z(f, \kappa) = 2C_l^T C_l f - 2\kappa C_h^T (g - C_h f) = 0$$

Wir bringen f auf die linke, g auf die rechte Seite:

$$(C_h^T C_h + \frac{1}{\kappa} C_l^T C_l) f = C_h^T g$$

Wir können nun wieder in die Faltung übersetzen:

$$f * (h^R * h + \frac{1}{\kappa} l^R * l) = h^R * g$$

Nun wenden wir darauf das Faltungstheorem an. Die Dimension nennen wir hier N_d, um sie vom Rauschen N zu unterscheiden:

$$\sqrt{N_d} \cdot \alpha_k(f) \cdot \left(\sqrt{N_d} |\alpha_k(h)|^2 + \sqrt{N_d} \frac{1}{\kappa} |\alpha_k(l)|^2 \right) = \sqrt{N_d} \cdot \overline{\alpha_k(h)} \alpha_k(g)$$

Wir lösen auf:

$$\alpha_k(f) \ = \frac{1}{\sqrt{N_d}} \frac{\overline{\alpha_k(h)} \alpha_k(g)}{|\alpha_k(h)|^2 + \frac{1}{\kappa} |\alpha_k(l)|^2} \tag{10.2}$$

Die nun noch ausstehende Ableitung der Langrangefunktion nach κ liefert die Ausgangsgleichung $\|g - h * f\|^2 = S$. Hier können wir jetzt die gefundene Lösung für f einsetzen und den noch fehlenden Wert für κ berechnen. In der Praxis wird man oft einen anderen Wert gehen: Da das Rauschen ohnehin nur grob bekannt ist und man hier auch "raten" beziehungsweise probieren müsste, kann man das κ auch direkt ausprobieren.

Ein sinnvoller LSI-Operator L_l könnte glatte Funktionen/Bilder auswählen. Dazu muss er bei örtlichen Änderungen im Bild ein hohes Signal liefern. Geeignet wären also Ableitungsfilter, zum Beispiel das Laplacefilter:

$$l = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Im einfachsten Fall wählen wir als LSI-Operator den identischen Operator $L_l = I$ oder $l = \delta$. Damit suchen wir unter allen zulässigen f dasjenige mit $\|\delta * f\|^2 = \|f\|^2 \rightarrow min$, das Bild mit der kleinsten Norm. Die Lösung im Frequenzbereich können wir sofort aufschreiben:

$$\alpha_k(f) = \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)}\alpha_k(g)}{|\alpha_k(h)|^2 + \beta} \quad \text{mit} \quad \beta = \frac{1}{\kappa} \cdot \frac{1}{N_d} \tag{10.3}$$

Diese Formel können wir für $\alpha_k(h) \neq 0$ auch anders schreiben

$$\alpha_k(f) = \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)}\alpha_k(g)}{|\alpha_k(h)|^2 + \beta} = \frac{1}{\sqrt{N_d}} \cdot \frac{\alpha_k(g)}{\alpha_k(h)} \cdot \left(\frac{|\alpha_k(h)|^2}{|\alpha_k(h)|^2 + \beta} \right) \quad \text{für} \quad \alpha_k(h) \neq 0$$

Da Formel 10.3 bei $\alpha_k(h) = 0$ den Wert Null ergibt, können wir formulieren

$$\alpha_k(f) = \begin{cases} \frac{1}{\sqrt{N_d}} \frac{\alpha_k(g)}{\alpha_k(h)} \left(\frac{|\alpha_k(h)|^2}{|\alpha_k(h)|^2 + \beta} \right) & \alpha_k(h) \neq 0 \\ 0 & \alpha_k(h) = 0 \end{cases} \tag{10.4}$$

Diese Formulierung zeigt uns die Nähe dieser Lösung zur inversen Filterung. Der bei der inversen Filterung berechnete Fourierkoeffizient $\alpha_k(f) = \frac{1}{\sqrt{N_d}} \frac{\alpha_k(g)}{\alpha_k(h)}$ wird durch den nachfolgenden Faktor verringert, wenn $\alpha_k(h)$ betragsmäßig klein ist, die Frequenz k bei der Faltung also fast ausgelöscht wurde. Das Rauschen ist dann im Vergleich groß und die Wiederherstellung würde vor allem das Rauschen verstärken.
Für die konkrete Anwendung wird man natürlich die Formel 10.2 oder 10.3 verwenden.
Dieses Restaurationsfilter liefert in der Praxis schon sehr gute Ergebnisse und wird als *Pseudoinvers-Filter* bezeichnet. Dieser Begriff könnte zu Verwechslungen mit der Pseudoinversen (9.4) führen, hat aber direkt damit nichts zu tun.

10.3. Wiener-Helstrom-Optimal-Filter

Wie nehmen als Bildentstehungsmodell wie bisher

$$g = h * f + N$$

an, die Punktverbreiterungsfunktion h ist gegeben. Wir wollen im folgenden auch f und g als Zufallsfelder auffassen und verwenden deshalb große Buchstaben, also

$$G = h * F + N$$

Für das Rauschen N machen wir einige Annahmen:

- Es sei $E(\{N\}) = 0$
- Das Rauschen sei signalunabhängig, also gilt $E\{F_i \cdot N_j\} = E\{F_i\} \cdot E\{N_j\}$

Aus diesen beiden Annahmen sieht man, dass dann $E(F * N) = 0$ sein muss.
Beim originalen Wiener-Filter (Norbert Wiener, 1942) wurde zunächst nur das Rauschen betrachtet. Erst später (1968) bezog Helstrom die Punktverbreiterungsfunktion h mit in seine Überlegungen ein.
Wir suchen einen unbekannten Operator T, der mit $\tilde{f} = Tg$ die Originalfunktion bestmöglich rekonstruiert, so dass

$$E\{\|TG - F\|^2\} \rightarrow Minimum$$

gilt. Auf Grund dieses Ansatzes ist das Wiener-Filter ein MMSE-Filter (**M**inimum **M**ean **S**quare **E**rror). Einen allgemeinen nichtlinearen Operator können wir nicht ohne weiteres angeben. Wir müssen den Operator einschränken und nehmen an, T sei ein linearer Operator. Wir können T als eine Matrix A beschreiben und müssen lösen

$$E\{\|F - A \cdot G\|^2\} \rightarrow Minimum$$

Als notwendiges Kriterium gilt

$$E\{(A \cdot G - F) \cdot G^T\} = 0$$

Wir setzen für G das Bildentstehungsmodell ein und erhalten

$$E\{A \cdot (C_h F + N) \cdot (F^T C_h^T + N^T)\} = E\{F(F^T C_h^T + N^T)\}$$

Nun multiplizieren wir einfach aus. Da der Erwartungswert von N mit Null angenommen wird, verschwinden danach einige Terme. Mit der Bezeichnung $COV_X := E\{XX^T\}$ erhalten wir dann

$$A = COV_F \cdot C_h^T \left[C_h \cdot COV_F \cdot C_h^T + COV_N\right]^{-1}$$

Neben der Punktverbreiterungsfunktion h benötigen wir also noch die Kovarianzmatrizen des Bildes COV_F und des Rauschens COV_N. Selbst wenn wir diese Kovarianzmatrizen kennen, wird deren Dimension für praktische Aufgabenstellungen doch gewaltig sein. Deshalb fordern wir für den Operator T außer der Linearität auch noch die Verschiebungsinvarianz: T sei ein LSI-Operator mit der Punktverbreiterungsfunktion w und damit $Tg = L_w g = w * g$. Wir müssen nun lösen

$$E\{\|F - w * G\|^2\} \rightarrow Minimum$$

Dies ist aber nichts anderes, als das lineare Gleichungssystem $E\{C_G \cdot w\} = E\{F\}$ im Sinne der kleinsten Quadrate zu lösen. In dieser Notation können wir also die üblichen Gaußschen Normalengleichungen benutzen:

$$E\{C_G^T C_G w\} = E\{C_G^T F\}$$

Dies übersetzen wir wieder in die Faltung

$$E\{G^R * G * w\} = E\{G^R * F\}$$

und setzen nun wieder die Modellgleichung für G ein:

$$\begin{aligned}\left\{h^R * h * E\{F^R * F\} + h^R * E\{F^R * N\} + h * E\{N^R * F\} + E\{N^R * N\}\right\} * w \;&= \\ = \; h^R * E\{F^R * F\} + E\{N^R * F\}&\end{aligned}$$

Auf Grund unserer Voraussetzungen an das Rauschen verschwinden folgende drei Terme:

$$h^R * E\{F^R * N\} \;,\; h * E\{N^R * F\} \;,\; E\{N^R * F\}$$

Daher ergibt sich die Bestimmungsgleichung für das Filter w:

$$\left\{h^R * h * E\{F^R * F\} + E\{N^R * N\}\right\} * w \;=\; h^R * E\{F^R * F\}$$

Da uns nicht das Filter w, sondern das geschätzte Bild $\hat{f} = w * g$ aus einer Realisierung g von G interessiert, falten wir beiden Seiten mit g:

$$\left\{h^R * h * E\{F^R * F\} + E\{N^R * N\}\right\} * \hat{f} = h^R * E\{F^R * F\} * g$$

Wenn wir nun das Faltungstheorem anwenden, erhalten wir

$$\begin{aligned}\alpha_k(\hat{f}) &= \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)} \cdot E\left(|\alpha_k(F)|^2\right) \alpha_k(g)}{|\alpha_k(h)|^2 E\left(|\alpha_k(F)|^2\right) \;+\; \frac{E(|\alpha_k(N)|^2)}{N_d}} \\ &= \frac{1}{\sqrt{N_d}} \cdot \frac{\alpha_k g \cdot \overline{\alpha_k(h)}}{|\alpha_k(h)|^2 + \frac{E(|\alpha_k(N)|^2)}{N_d \cdot E(|\alpha_k(F)|^2)}}\end{aligned} \tag{10.5}$$

Wir benötigen also die mittleren Autokorrelationsfunktionen des Originals und des Rauschens bzw. die mittleren Leistungsspektren des Originalsignals und des Rauschens. Für das Rauschen wird oft weisses Rauschen angenommen. In diesem Falle ist

$$E\{N^R * N\} \;=\; \delta N_d \sigma^2 \;\rightarrow\; |\alpha_k(N)|^2 \;=\; \sigma^2$$

Die Autokorrelation des Originalsignals $E\{(F^R * F)_n\} \;=\; E\{\sum_l F_l F_{l-n}\}$. Da uns diese im Allgemeinen nicht zur Verfügung steht, müssen wir diese aus dem unscharfen Bild g schätzen. Da hier die Stichprobe nur ein Element enthält, bestehen Zweifel an der Zuverlässigkeit der Schätzung. Wenn wir aber bedenken, dass ein Summand $(K_F)_n = E\{F_k \cdot F_{k-n}\}$ nur noch von n abhängt, und nicht mehr von k, dann könnten wir die Autokorrelation tatsächlich auch aus einem Bild schätzen, da wir für jedes dieser Produkte genug Stichproben zur Verfügung haben. In der Stochastik werden solche Zufallsfelder oder Prozesse als **stationär im erweiterten Sinne** bezeichnet. Es bleibt aber immer noch das Problem, dass wir nur das unscharfe Bild für die Schätzung der Autokorrelation verwenden können. Hier bietet sich an, einen iterativen Ansatz zu verfolgen: Wenn nach der Restauration des Bildes mit der Autokorrelation des unscharfen Bildes das Bild verbessert ist, kann man das restaurierte Bild zur erneuten Schätzung der Autokorrelation benutzen und erneut das Bild restaurieren. Dazu gibt es in der Literatur zahlreiche Varianten dieser iterativen Wiener-Entfaltung.

10.4. Richardson-Lucy-Algorithmus

Im folgenden sollen einige statistische Betrachtungen angestellt werden. Wir gehen wieder vom Modell

$$g = h * f + N$$

aus. Wenn wir nun annehmen, alle N_k seien unabhängig, normalverteilt mit dem Erwartungswert Null und der gleichen Varianz σ, dann könnten wir folgende Likelihoodfunktion

$$L(f) = \prod_{k=0}^{N-1} e^{-\frac{(g_k-(h*f)_k)^2}{2\sigma^2}}$$

zur Schätzung von f aufstellen. Statt $L(f))$ zu maximieren, minimieren wir $-ln(L(f))$ und sehen sofort, dass dann

$$\|g - h * f\|^2 \;\rightarrow\; Minimum$$

gelten muss. Dieses Ergebnis beschreibt genau die inverse Filterung 9.5. Verwunderlich ist dies nicht: Für eine Normalverteilung ist der Erwartungswert auch der wahrscheinlichste Wert, so dass das Rauschen keinen Einfluss auf das Ergebnis hat.
Aber vielleicht liefert dieses Vorgehen ein vernünftiges Ergebnis, wenn N nicht normalverteilt ist. Dazu nehmen wir wieder an, die Rauschwerte N_k seien unabhängig und es liege Poisson-Rauschen vor. Die Grauwerte in g unterliegen der **Poissonverteilung** mit dem Erwartungswert $(h * f)_k$. Eine Zufallsvariable X ist poissonverteilt, wenn

$$P(X = l) \;=\; \frac{\lambda^l e^{-\lambda}}{l!}$$

gilt. λ ist dabei der Erwartungswert von X. Da wir von ganzzahligen Grauwerten g_k ausgehen, ist diese Annahme durchaus möglich. Wir bilden wieder die Likelihood-Funktion

$$L(f) \;=\; \prod_{k=0}^{N-1} \frac{(h * f)_k{}^{g_k} e^{-(h*f)_k}}{g_k!} \;\rightarrow\; Maximum$$

Wir bilden wieder den Logarithmus

$$\ln(L(f)) \;=\; \sum_{k=0}^{N-1} [g_k \ln(h * f)_k) - (h * f)_k - \ln(g_k!)] \;=$$

$$= \sum_{k=0}^{N-1} \left[g_k \ln(\sum_{j=0}^{N-1} h_{k-j} f_j) - \sum_{j=0}^{N-1} h_{k-j} f_j - \ln(g_k!) \right]$$

Wir bilden wie üblich die partiellen Ableitungen

$$\frac{\partial \ln(L(F))}{\partial f_l} = \sum_{k=0}^{N-1} \frac{g_k}{\sum_{j=0}^{N-1} h_{k-j} f_j} h_{k-l} - \sum_{k=0}^{N-1} h_{k-l} = 0 \quad \text{mit} \quad l = 0, ..., N-1$$

Diese Gleichung können wir kompakt wieder als Faltung schreiben:

$$\sum_{k=0}^{N-1} \frac{g_k}{(h*f)_k} h^R_{l-k} = ((\frac{g}{h*f}) * h^R)_l = \sum_{k=0}^{N-1} h_{k-l} \quad \text{mit} \quad l = 0, ..., N-1$$

In dieser Schreibweise bedeutet der Quotient eine pixelweise Operation. Diese nichtlineare Gleichung ist nun für f zu lösen. Dazu wird in der Literatur ein einfaches iteratives Vorgehen vorgeschlagen:

$$f_l^{(n+1)} = \frac{1}{\left(\sum_{k=0}^{N-1} h_{k-l}\right)} f_l^{(n)} ((\frac{g}{h * f^{(n)}}) * h^R)_l \quad \text{mit} \quad l = 0, ..., N-1 \tag{10.6}$$

Als Startbild $f^{(0)}$ kann g, aber beispielsweise auch ein konstantes Bild $f_{x,y} = c$ genutzt werden.
Diese Iteration wird Richardson-Lucy-Algorithmus genannt, siehe [34] und [25]. Es ist auffällig, dass dieser Algorithmus oft in der Astronomie verwendet wird. Oft wird dieser Algorithmus auch als EM-Algorithmus mit Poisson-Statistik bezeichnet. In der Literatur gibt es eine Menge von Arbeiten, die sich mit Konvergenzuntersuchungen befassen und Vergleiche zu den anderen Restaurationsmethoden anstellen. Die Iteration konvergiert sehr langsam und ist daher numerisch sehr aufwändig.
Interessanterweise wird oft berichtet, dass die Iteration rechtzeitig abgebrochen werden sollte. Das ist erstaunlich, da der Grenzwert die nach dem Ansatz optimale Rekonstruktion darstellen sollte. Die berichteten guten Eigenschaften könnten somit gar nicht unbedingt beim Maximum-Likelyhood-Ansatz mit der Poisson-Verteilung liegen, sondern im Iterationsverhalten von 10.6.

Teil V.

Abtasttheoreme

11. Das Abtasttheorem

Bei vielen Aufgabenstellungen sind die diskreten Funktionen eine Approximation einer analogen Funktion. So erfordern alle numerischen Berechnungen eine Digitalisierung/Abtastung, obwohl meist eigentlich die analoge Funktion betrachtet werden sollte. Deshalb ist es wichtig entscheiden zu können, ob die digitale Funktion die analoge Funktion exakt oder wenigstens genau genug beschreibt. Dies soll die Fragestellung in diesem Abschnitt sein.
Das Abtasttheorem beschreibt, unter welchen Bedingungen eine diskrete Funktion eine analoge Funktion vollständig beschreibt, so dass diese aus den abgetasteten Werten wiederhergestellt werden kann.

11.1. Trigonometrische Interpolation

Wir wollen uns zunächst einen Weg ansehen, wie aus einer diskreten Funktion eine analoge Funktion erzeugt werden kann. Fordert man, dass die analoge Funktion an den Stützstellen die Werte der diskreten Funktion haben soll, handelt es sich um eine Interpolation.
Gegeben sei eine diskrete Funktion f mit endlichem Definitionsbereich $0 \leq k < N$. Um den Bezug zur analogen Funktion herzustellen, ordnen wir den diskreten Werten des Definitionsbereichs Stützstellen im Reellen zu:

$$n \to x_n = n \cdot \Delta x \quad \text{mit} \quad n = 0, 1, ..., N-1$$

Wenn die diskrete Funktion f die Periode N hat, dann muss die Periode der analogen Funktion $X = N \cdot \Delta x$ sein und es gilt:

$$\frac{x_n}{X} = \frac{n}{N}$$

Von unserer Interpolation erwarten wir, dass die interpolierte Funktion f_a an der Stützstellen den Wert der diskreten Funktion haben

$$f_a(x_n) = f_n$$

Von der diskreten Funktion f berechnen wir die Fourierkoeffizienten

$$\alpha_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l e^{-2\pi i k \frac{l}{N}}$$

Offensichtlich ist nun

$$f_a(x_n) = f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}}$$

Wir wollen nun diese Formel für die Funktionswerte an den Stützstellen x_n auf alle x erweitern. Dazu ersetzen wir einfach x_n durch x und $\frac{n}{N}$ durch $\frac{x}{X}$:

$$f_a(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{x}{X}}$$

Im Diskreten konnten wir irgendein Intervall der Länge N für k wählen, weil die Basisfunktionen φ_k und φ_{k+N} identisch sind. Gehen wir aufs Analoge über, stimmt das nicht mehr. Die Interpolation ist nun von der Wahl des Intervalles für k abhängig. Wir nennen den ersten Wert k_0 und erhalten für die trigonometrische Interpolation:

$$f_a^{k_0}(x) = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{x}{X}} \tag{11.1}$$

Jeder Wert für k_0 führt zu einer korrekten Interpolation. Gehen wir allerdings von reellwertigen Funktionen aus, erwarten wir auch eine Interpolation mit reellen Funktionswerten. Wir wissen, dass dazu für die Fourierkoeffizienten $\alpha_{-k} = \overline{\alpha_k}$ gelten muss. Deshalb müssen wir k_0 so wählen, dass das Intervall symmetrisch zu 0 liegt. Bei ungeradem N wählen wir also $k_0 = -\frac{N-1}{2}$. Bei geradem N haben wir einen Fourierkoeffizient zusätzlich. Hier besitzt $\alpha_{\frac{N}{2}}$ kein negatives Gegenstück, denn $\alpha_{-\frac{N}{2}} = \alpha_{-\frac{N}{2}+N} = \alpha_{\frac{N}{2}}$ ist derselbe Fourierkoeffizient. Da auch hier für reelle Funktionen $\alpha_{-\frac{N}{2}} = \overline{\alpha_{\frac{N}{2}}}$ gelten muss, kann dieser Fourierkoeffizient nur rein reell sein, sein Imaginärteil ist Null. Deshalb ist $k_0 = -\frac{N-1}{2}$ auch hier sinnvoll, wenn man unter der Division die ganzzahlige Division versteht.

11.2. Abtasttheorem im endlichen Intervall

Es sei nun eine analoge Funktion $f_a(x)$ im Intervall $[0, X]$ gegeben. Die AFT lautet dann

$$\begin{aligned} f_a(x) &= \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \beta_k e^{+2\pi i k \frac{x}{X}} \\ \beta_k &= \frac{1}{\sqrt{X}} \int_0^X f_a(x) e^{-2\pi i k \frac{x}{X}} dx \end{aligned}$$

Wir tasten nun die analoge Funktion $f_a(x)$ an den äquidistanten Stützstellen $x_n = \frac{n}{N} X$ ab und erhalten damit die diskrete Funktion $f_d(x_n)$ mit

$$f_a(x_n) \;=\; f_d(x_n) \quad n = 0, 1..., N-1$$

Wir stellen jetzt die Frage, ob und wann das Interpolationpolynom gleich der Originalfunktion ist. Wir verwenden dazu die Darstellung im Frequenzbereich.

$$\begin{aligned} f_a(x) &= f^{k_0}(x) \\ \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \beta_k e^{2\pi i k \frac{x}{X}} &= \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{x}{X}} \end{aligned}$$

Auf beiden Seiten steht nun eine Entwicklung nach dem Orthonormalsystem der Fouriertransformation. Gleichheit kann also nur dann bestehen, wenn die Koeffizienten die gleichen sind und wir erhalten:

$$\frac{1}{\sqrt{X}} \beta_k = \begin{cases} \frac{1}{\sqrt{N}} \alpha_k & k_0 \leq k < k_0 + N \\ 0 & \text{sonst} \end{cases}$$

Daraus können wir jetzt das Abtasttheorem formulieren:
Abtasttheorem: Ist eine analoge Funktion bandbegrenzt in dem Sinne, dass $\beta_k = 0$ für $k < k_0 \vee k \geq k_0 + N$, dann kann die analoge Funktion $f_a(x)$ vollständig aus der diskreten Funktion der Abtastwerte $f_d(x_n)$ rekonstruiert werden und ist gleich dem trigonometrischen Interpolationspolynom $f_{k_0}(x)$.
Wir haben schon bei der trigonometrischen Interpolation gesehen, dass für reellwertige Funktionen nur ein zu Null symmetrischer Frequenzbereich sinnvoll ist. Nehmen wir dazu eine Bandbegrenzung in der folgenden Form an:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-K}^{+K} \beta_k e^{2\pi i k \frac{x}{X}} \tag{11.2}$$

Die bandbegrenzte Funktion $f(x)$ hat dann $N = 2K + 1$ von Null verschiedene Fourierkoeffizienten und die größte Frequenz ist $\nu_g = \frac{K}{X}$. Die Abtastfrequenz ist $\nu_a = \frac{N}{X}$. Da eine höhere Zahl der Abtastpunkte nicht schadet, erhalten wir daraus die übliche Formulierung $\nu_a > 2\nu_g$.
Interpretation

- Ist das analoge Signal bandbegrenzt in dem Sinne, dass genau N aufeinanderfolgende Fourierkoeffizienten des analogen Signals relevant sind und die anderen den Wert Null besitzen, dann reicht die Kenntnis dieser N komplexen Fourierkoeffizienten zur Rekonstruktion der analogen Funktion durch trigonometrische Interpolation.

- Um die N relevanten Fourierkoeffizienten zu ermitteln, muss die Funktion an M Stellen abgetastet werden, wobei $M \geq N$ sein muss. Die diskrete Funktion der Abtaststellen beschreibt die analoge Funktion vollständig. Die Lage des Frequenzbandes ist beliebig in der Art $k_0 \leq k < k_0 + N$.

- Für reellwertige Signale muss das Frequenzband symmetrisch zur Null liegen. Für die Abtastfrequenz ergibt sich $\nu_a > 2\nu_g$ mit ν_g als (obere) Grenzfrequenz des analogen Signals.

11.3. Abtasttheorem für das unendliche Bildmodell

Nun betrachten wir analoge Funktionen $f_a(x)$ im Intervall $-\infty < x < +\infty$ und verwenden die Fourierintegraltransformation IFT (6.5):

$$\alpha_a(\nu) = \int_{-\infty}^{\infty} f_a(x) e^{-2\pi i \nu x} dx$$

$$f_a(x) = \int_{-\infty}^{\infty} \alpha_a(\nu) e^{+2\pi i \nu x} d\nu$$

Auch die abgetastete diskrete Funktion ist unendlich und wir verwenden die unendliche diskrete Fouriertransformation (6.7):

$$\alpha(\omega) = \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} f_k e^{-ik\omega} \quad , \quad f_k = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \alpha(\omega) e^{+ik\omega} \, d\omega$$

Wir nehmen nun an, die diskrete Funktion ist Resultat der Abtastung der analogen Funktion f_a mit der Schrittweite Δx. Wenn wir aus dem Spektrum dieser diskreten Funktion das Spektrum der analogen Funktion berechnen können, können wir die analoge Funktion aus den Abtastwerten vollständig rekonstruieren. Dies ist die gleiche Idee wie beim Abtasttheorem im endlichen Intervall. Wir müssen deshalb die diskrete Darstellung an die analoge Darstellung anpassen, indem wir das ganzzahlige n durch die Stützstellen $x_n = n \cdot \Delta x$ ersetzen.

$$f_d(x_n) = f_d(n \cdot \Delta x) = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \alpha_d(\omega_d) e^{+i\omega_d \frac{x_n}{\Delta x}} \, d\omega_d$$

Wenn wir die analogen und diskreten Basisfunktionen einander zuordnen wollen, müssen wir $\nu = \frac{\omega_d}{2\pi\Delta x}$ setzen. Wir substituieren $\omega_d \to 2\pi\Delta x\nu$:

$$f_d(x_n) = \frac{1}{\sqrt{2\pi}} \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} \alpha_d(2\pi\Delta x\nu) e^{+2\pi i \nu x_n} \, 2\pi\Delta x \, d\omega = \sqrt{2\pi}\Delta x \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} \alpha_d(2\pi\Delta x\nu) e^{+2\pi i \nu x_n} \, d\omega$$

Wenn wir statt $x_n = n\Delta x$ beliebige x zulassen, haben wir die Entsprechung zur trigonometrischen Interpolation in Gleichung 11.1. Wir fordern jetzt wieder, dass die Interpolation und die analoge Funktion übereinstimmen:

$$f_a(t) = \int_{-\infty}^{\infty} \alpha_a(\nu) e^{+2\pi i \nu x} d\nu = f_d(t) = \sqrt{2\pi}\Delta x \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} \alpha_d(2\pi\Delta x\nu) e^{+2\pi i \nu x} d\nu$$

Wieder können wir die Koeffizienten vergleichen und erhalten:

$$\alpha_a(\nu) = \begin{cases} \sqrt{2\pi}\Delta x \alpha_d(2\pi\Delta x\nu) & -\frac{1}{2\Delta x} \leq \nu \leq \frac{1}{2\Delta x} \\ 0 & \text{sonst} \end{cases}$$

Die Gleichheit können wir also genau dann erreichen, wenn die Fourierkoeffizienten ausserhalb von $-\frac{1}{2\Delta x} \leq \nu \leq \frac{1}{2\Delta x}$ gleich Null sind, die analoge Funktion also bandbegrenzt ist.

Bemerkung 1: Bei einer gegebenen Grenzfrequenz ν_o und $-\nu_o \leq \nu \leq \nu_o$ erhalten wir $\Delta x \leq \frac{1}{2\nu_o}$ oder mit der Abtastfrequenz $\nu_s = \frac{1}{\Delta X}$ die Formulierung $\nu_s \geq 2\nu_o$. Dies entspricht der üblichen Formulierung des Abtasttheorems, wenn wir eine zum Koordinatenursprung symmetrische Bandbegrenzung haben.

Bemerkung 2: Wir haben die Bandbegrenzung symmetrisch zum Koordinatenursprung gewählt. Dies ist bei reellwertigen Funktionen sinnvoll, da hier $\alpha(-\nu) = \overline{\alpha(\nu)}$ sein muss. Bei komplexwertigen Funktionen können wir die Bandbegrenzung auf ein beliebiges Intervall $\nu_1 \leq \nu \leq \nu_2$ beziehen. Daraus folgt $\Delta x \leq \frac{1}{\nu_2 - \nu_1}$.

Bemerkung 3: Die korrespondierenden Fourierkoeffizienten sind hier $\alpha_a(\nu)$ und $\alpha_d(2\pi\Delta x\nu)$. Die unterschiedlichen Argumente ergeben sich aus dem bei der unendlichen diskreten Fouriertransformation willkürlich gewählten Intervall für das Spektrum.

11.4. Modell einer realen Abtastung

Bei den bisherigen Betrachtungen zur Abtastung sind wir von einer idealen Abtastung der Funktion f ausgegangen, bei der die diskrete Funktion g den Werten der analogen Funktion f an äquidistanten Stützstellen entspricht:

$$g_n = f(x_n) = f(n\Delta x)$$

Dies ist bei der realen Abtastung so nicht zu finden: In realen Systemen zu Digitalisierung repräsentiert jeder Abtastwert nicht nur einen [Zeit-]Punkt in der originalen Funktion. Bildsensoren bestehen beispielsweise aus einer Matrix von Pixeln, die jeweils das Licht erfassen, was auf ihre Sensorfläche fällt. Betrachtet man die Sensorzelle in Koordinatenursprung, so lässt sich ihre Empfindlichkeit durch eine Funktion $e(x, y)$ beschreiben. Zur Vereinfachung der Beschreibung wollen wir dies im folgenden eindimensional als $e(x)$ schreiben. Ist $f(x)$ die Originalfunktion und g_k die digitalisierte Funktion, so lässt sich das Pixel im Ursprung beschreiben zu

$$g_0 = \int_{-\infty}^{+\infty} e(\xi) f(\xi) d\xi$$

Für eine Zelle an der Stelle x_k müssen wir die Empfindlichkeitsfunktion dorthin verschieben:

$$g_k = \int_{-\infty}^{+\infty} e(\xi - x_k) f(\xi) d\xi$$

Durch Spiegeln der Empfindlichkeitsfunktion wird aus dem Ausdruck eine Faltung:

$$g_k = \int_{-\infty}^{+\infty} e^R(x_k - \xi) f(\xi) d\xi = (e^R * f)(x_k)$$

Eine reale Abtastung, bei der die Empfindlichkeitsfunktion für alle Abtastpunkte gleich ist, lässt sich also beschreiben als Faltung der Originalfunktion f mit der Empfindlichkeitsfunktion e und nachfolgende ideale Abtastung am Punkt x_k.

11.5. Unregelmäßige Abtastpunkte

Im Kern sagt das Abtasttheorem, dass wenn eine Funktion nur N von Null verschiedene Fourierkoeffizienten hat, N Stützstellen bei der Abtastung genügen. Damit sind die Fourierkoeffizienten berechenbar und die analoge Funktion ist rekonstruierbar. Bisher haben wir äquidistante Stützstellen vorausgesetzt und konnten die Fourierkoeffizienten mittels der DFT effizient berechnen. Würden denn auch N beliebige Stützstellen genügen? Hier können wir die DFT nicht so einfach zu rate ziehen. Wir verwenden eine Fourierreihe:

$$f(x) = \frac{1}{X} \sum_{n=-\infty}^{\infty} \beta_n e^{2\pi n \frac{x}{X}}$$

Da nur N Fourierkoeffizienten ungleich null sind, können wir den Summationsbereich auf N Summanden begrenzen. (Wir verwenden hier die Indizes 0 bis $N-1$, auch wenn es oft andere Fourierkoeffizienten sein werden, die von Null verschieden sind.)

$$f(x) = \frac{1}{X} \sum_{n=0}^{N-1} \beta_n e^{2\pi n \frac{x}{X}}$$

Von N Stützstellen x_k kennen wir die zugehörigen Funktionswerte $f(x_k)$. Die Bestimmung der β_n stellt nun die Lösung eines linearen Gleichungssystems dar.

$$y_k = f(x_k) = \frac{1}{X} \sum_{n=0}^{N-1} \beta_n e^{2\pi n \frac{x_k}{X}} \quad \text{mit} \quad k = 0, 1, .., N-1$$

Wir führen die Basisfunktionen φ_n auf die Basisfunktion φ_1 zurück:

$$\varphi_n(x) = e^{2\pi n \frac{x}{X}} = e^{2\pi \frac{x}{X} \cdot n} = \varphi_1(x)^n$$

In der Matrix-Schreibweise haben wir jetzt

$$y = A \cdot \beta$$

mit den Matrix A

$$A = \frac{1}{\sqrt{X}} \begin{pmatrix} 1 & \varphi_1(x_0) & \varphi_1(x_0)^2 & \cdots & \varphi_1(x_0)^{N-1} \\ 1 & \varphi_1(x_1) & \varphi_1(x_1)^2 & \cdots & \varphi_1(x_1)^{N-1} \\ & & & \vdots & \\ 1 & \varphi_1(x_{N-1}) & \varphi_1(x_{N-1})^2 & \cdots & \varphi_1(x_{N-1})^{N-1} \end{pmatrix}$$

Diese Matrix ist eine Vandermonde-Matrix: In der zweiten Spalte stehen die Funktionswerte von φ_1 an den Stellen x_n, in den anderen Spalten die Potenzen dieser Werte. Die Determinante dieser Matrix wird nur dann Null, wenn zwei Werte $\varphi_1(x_k)$ gleich sind. Das würde nur dann passieren, wenn die x_k zusammenfallen. Damit ist das Gleichungssystem im Normalfall $x_k \neq x_j$ bei $k \neq j$ immer lösbar. Es reichen also N beliebige Punkte aus. Leider ist in diesem Fall die Bestimmung aber nicht so einfach wie bei der DFT mit äquidistanten Punkten, da die Matrix nicht orthogonal ist.

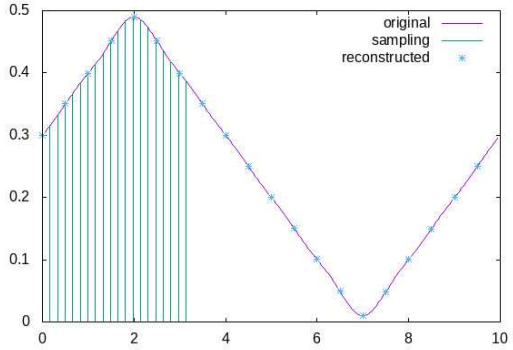


Abbildung 11.1.
20 Abtastpunkte im linken Drittel
Rekonstruktion mit geringem Fehler

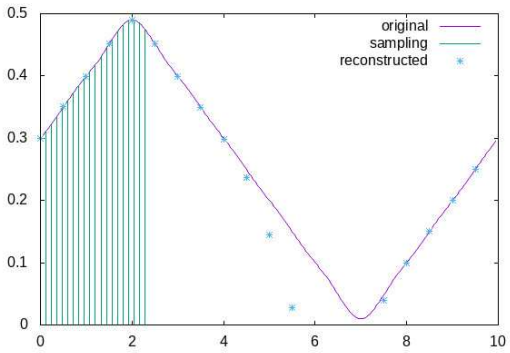


Abbildung 11.2.
20 Abtastpunkte im linken Viertel
Rekonstruktion mit Fehlern

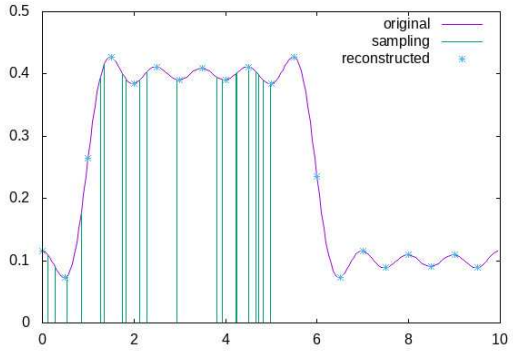


Abbildung 11.3.
20 zufällige Abtastpunkte
Rekonstruktion mit geringem Fehler

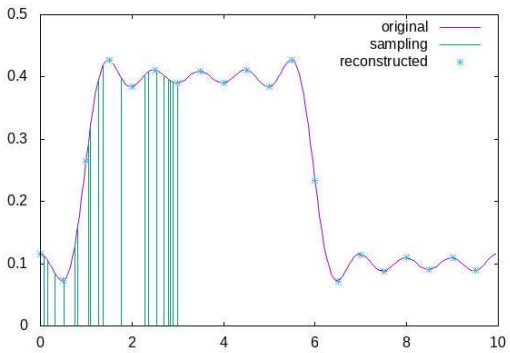


Abbildung 11.4.
20 zufällige Abtastpunkte
Rekonstruktion mit Fehlern

Dieses Ergebnis ist verblüffend. Stellen wir uns ein Bild vor, welches bandbegrenzt ist und wir tasten die notwendige Anzahl von Werten alle in einer Ecke ab. Theoretisch könnten wir das Bild aus dieser Information rekonstruieren. Da im analogen Bild die Abtastpunkte beliebig dicht liegen können, bedeutet das, dass jedes noch so kleine Gebiet die gesamte Bildinformation enthält. Wir sehen, dass die Bandbegrenzung offensichtlich eine sehr wesentliche Einschränkung darstellt.
Dies ist jedoch ziemlich theoretisch: Das Gleichungssystems ist in diesem Fall schlecht konditioniert und die Lösung instabil. Die Lösung wird stabil, wenn die Abtastpunkte gleichmäßiger im Intervall verteilt werden, kleinere Lücken lassen sich überbrücken. Der stabilste Fall ist aber, wenn man äquidistante Abtastpunkte über das ganze Intervall verteilt. Die Lösung des linearen Gleichungssystems entspricht dann der klassischen DFT.

12. Anwendungen des Abtasttheorems

12.1. Digitalisierung

Bei der Digitalisierung von analogen Signalen ist offensichtlich das Abtasttheorem unmittelbar wichtig.
Um ein analoges Bild oder allgemein ein analoges Signal im Rechner abzuspeichern, müssen wir das analoge Signal in die diskrete Form überführen. Dazu benötigen wir eine **Rasterung** des Definitionsbereiches und eine **Quantisierung** des Wertebereiches, beides zusammen nennen wir **Digitalisierung** des analogen Signals.
Wenn wir keine zusätzlichen Bemerkungen machen, dann erfolgt die Rasterung immer äquidistant. Zweidimensional bedeutet dies die Abtastung im Quadratgitter. Nach der Abtastung nummerieren wir abstrakt die Abtastpunkte mit ganzen Zahlen durch. f_n ist somit nicht unbedingt der Funktionswert an der Stelle n der analogen Funktion $f_a(x)$, es ist der n-te abgetastete, diskrete Funktionswert. Bei äquidistanter Abtastung im Abstand Δx gilt folglich

$$f_n = f_a(x_n) = f_a(n \cdot \Delta x)$$

Das Abtasttheorem sagt uns, dass bei Abtastung kein Verlust auftritt, wenn die Grenzfrequenz des Signals ν_g weniger als halb so groß wie die Abtastfrequenz ν_a ist. Die halbe Abtastfrequenz $\nu_{Nyquist} = \frac{1}{2}\nu_{ab}$ wird auch **Nyquist-Frequenz** genannt, so dass gilt $\nu_{Nyquist} \geq \nu_g$.
Eine andere Formulierung bezieht sich auf die Wellenlänge oder Periode der Schwingung mit der höchsten Frequenz: Bezüglich der Periode der Schwingung mit der Grenzfrequenz muss man mindestens 2 Abtastpunkte wählen.

12.2. Beispiele zur Digitalisierung

Wir wollen zwei einfache Beispiele der Digitalisierung betrachten:

Beispiel 1: Für ein gewöhnliches analoges Fernsprechsignal zur Übertragung von Sprache reicht eine Übertragung des Frequenzbereiches von 300 bis 3400 Hertz aus. Wenn wir das Signal entsprechend filtern, liegt also eine Grenzfrequenz von 3400 Hertz vor. Zur Digitalisierung müssen wir das Signal im Abstand von

$$\Delta t \leq \frac{1}{2 \cdot 3400} = 147 \cdot 10^{-6} s = 147 \mu s$$

abtasten, wenn keine Verluste auftreten sollen. Aus Sicherheitsgründen wählt man einen Abstand von $125\mu s$. Als eigentliche Abtastfrequenz erhalten wir

$$\nu_{ab} = 2 \cdot \nu_{Nyquist} = \frac{1}{125\mu s} = 8000Hz$$

Dies ist die beim digitalen Telefonstandard ISDN verwendete Abtastfrequenz. Wenn genügend Übertragungskapazität zur Verfügung steht, wird heute oft eine größere Bandbreite und damit Abtastrate verwendet.
Beispiel 2: Ein analoges Video-Signal nach der Fernsehnorm CCIR, wie es in Deutschland üblich war, hat folgende Eckdaten:

- Interlace Mode 2 : 1
- Halbbildfrequenz 50 Hz
- Vollbildfrequenz 25 Hz
- Zeit zum Aufbau eines Vollbildes 40 ms
- Anzahl Zeilen 625
- Sichtbare Zeilen 576
- Verhältnis horizontale zu vertikale Auflösung 4 : 3.

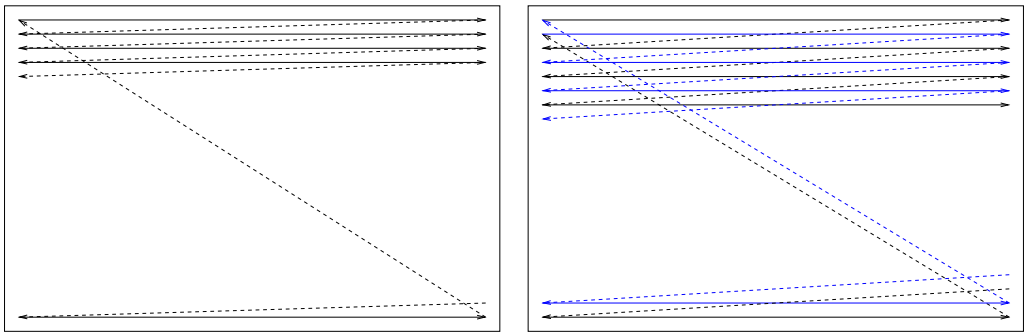


Abbildung 12.1.
Analoge Abtastung eines Fernsehbildes - links: progressive scan, rechts: interlaced

Das Bild wird zeilenweise abgetastet. Bei der Bewegung entlang der Zeile entsteht eine analoge Funktion, deren Zeitabhängigkeit die Änderung des Bildes bezüglich der x-Achse darstellt. In y-Richtung springt die Abtastung zu jeweils nächsten Zeile. Es handelt sich somit nicht um eine echte analoge 2D-Funktion. Bezüglich der y-Richtung liegt bereits eine Rasterung vor.
Beim Übergang zur digitalen Verarbeitung wurde das analoge Signal digitalisiert. Gehen wir einmal ideal von den 625 Zeilen aus. Wenn wir abtasten mit dem Ziel quadratische Pixel

zu erhalten, so müssen wir $625 \cdot \frac{4}{3} = 835$ Bildpunkte pro Zeile abtasten. Damit müssen wir pro Sekunde

$$25 \cdot \ 625 \cdot 835 \ \approx \ 13 \cdot 10^6$$

Signale abtasten oder übertragen. Die Abtastfrequenz muss also 13 MHz betragen. Die CCIR-Norm beschränkt die Bandbreite des Video-Signals aber auf 5 MHz. Eine Abtastung mit 10 MHz ist damit ausreichend, um das analoge Signal reproduzieren zu können. Daher ist die vertikale Auflösung des Fernsehbildes höher als die horizontale Auflösung. Die Norm CCIR-601 legt für die Digitalisierung eines derartigen Signales eine Abtastfrequenz von 13,5 MHz fest, was damit sicher ausreicht. Moderne Fernsehnormen definieren eine digitale Übertragung der Fernsehsignale ohne die Digitalisierung eines analogen Signals.

Interlace-Modus (Zeilensprungverfahren) bedeutet, dass bei der zeilenweisen Abtastung zunächst die ungeraden, im zweiten Durchlauf dann die geraden Zeilen abgetastet werden (Abbildung 12.1 rechts). Da dann nach der halben Zeit bereits ein Halb-Bild vorliegt, verdoppelt sich die für das Flimmern wirksame Bild-Frequenz und es entsteht ein ruhigerer Bildeindruck, wenn wie beim klassischen analogen Fernsehen das Bild unmittelbar auf den Bildschirm geschrieben wird. Halbbilder haben die halbe Auflösung in Y-Richtung. Wollen wir aber ein Vollbild analysieren, so setzt sich dieses aus zwei Halbildern zusammen. Diese wurden aber zeitlich versetzt aufgenommen und passen deshalb nicht exakt zusammen, wenn sich im Bild etwas bewegt hat. Die Halbbilder sind versetzt und es entsteht eine kammartige Struktur. Dies wird dann oft durch Interpolation ausgeglichen, man nennt dies **Deinterlacing**. Praktisch wird dadurch die Auflösung verringert. Aufnahmen, die direkt auf Vollbildern beruhen, nennt man **Progressive Scan**. Beide Abtastmethoden spielen auch beim digitalen Fernsehen eine Rolle und werden in den Fernsehstandards durch einen nachgestellten Buchstaben ausgedrückt: 720p ist ein üblicher **Progressive Scan**-Standard, bei 1080i hingegen erfolgt die Abtastung **interlaced**.
Typisch ist heute die direkte Abtastung über die geometrisch angeordneten Sensorzellen, in der Regel CCD- oder CMOS-Sensoren. Damit sind Interlaced-Modus und progressive scan-Modus nur noch Fragen der Abfrage der Sensoren und der Übertragung des Signals.

12.3. Über- und Unterabtastung

Wir kehren noch einmal zurück zum Abtasttheorem. Wenn wir weniger Abtastpunkte als im Theorem angegeben verwenden, dann spricht man von **Unterabtastung**. Wir können dann das Signal nicht mehr vollständig rekonstruieren. Dies fällt insbesondere bei periodischen Mustern auf: Es entstehen Strukturen, die es im Originalbild nicht gibt. Dies nennt man **Aliasing**. In der Abbildung 12.2 ist eine simulierte Abtastung des Siemens-Sternes zu sehen, es entstehen Aliasing-Effekte, die an die Computer-Grafik erinnern. Auch bei der Bearbeitung von Bildern kann Subsampling auftreten, beispielsweise bei der Verkleinerung, also einer nachträglichen Reduktion der Zahl der Abtastpunkte (12.4.1).

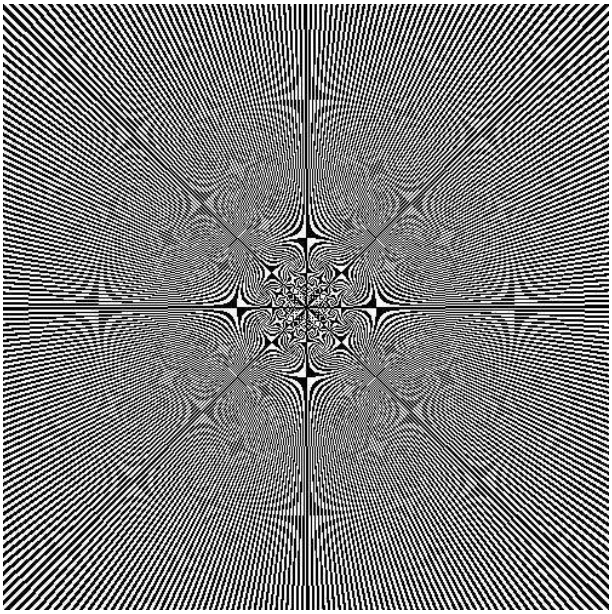


Abbildung 12.2.: Siemens-Stern mit Aliasing-Effekten

Überabtastung ist die Abtastung mit einer Abtastfrequenz, die größer als die nach dem Abtasttheorem erforderliche Abtastfrequenz ist. Dies ist offensichtlich kein Gewinn an Information. Technische Erfordernisse können die Überabtastung erfordern, wie zum Beispiel die gegebene Auflösung eines Monitors, die die Vergrößerung eines Bildes, also die Darstellung mit mehr Abtastpunkten erfordern (12.4.2).

12.4. Änderung der Auflösung

12.4.1. Verkleinern von Bildern (Shrinking)

Eine unmittelbare Anwendung des Abtasttheorems finden wir beim Verkleinern von digitalen Bildern. Wir nehmen einmal ein eindimensionales Bild der Größe N an, dann haben wir genau N DFT-Koeffizienten. Nach der Verkleinerung auf die halbe Größe hat das neue Bild die Dimension $N/2$. Wenn im Originalbild die hohen Frequenzen, also die oberen $N/2$ DFT-Koeffizienten, nicht Null sind, geht eindeutig Information verloren. Haben wir die Verkleinerung erreicht, indem wir jedes zweite Pixel weglassen, liegt eindeutig eine Unterabtastung vor. Somit werden feine Strukturen verschwinden und Aliasing-Effekte auftreten. Um diese zu vermeiden, müssten wir vorher das Bild mit einem Tiefpass filtern, so dass die hohen Frequenzen nicht mehr vorhanden sind. Bei der Verkleinerung tritt dann kein weiterer Verlust auf. In jedem Fall wird das Bild dadurch aber unschärfer und bei einem "idealen" Tiefpass treten bekannterweise auch Ringing-Artefakte auf. Letztlich können wir einen Informationsverlust nicht vermeiden. Ein optimales Vorgehen gibt es hier nicht. Ist das Original zum Beispiel nahezu ein Binärbild ohne sehr kleine Strukturen, kann selbst das

einfache Weglassen ein subjektiv besseres Bild erzeugen, als das mittels Tiefpass erzeugte.

Abbildung 12.3.
Originalbild

Abbildung 12.4.
Verkleinert mit Tiefpassfilterung
Darstellung vergrößert

Abbildung 12.5.
Verkleinert durch Weglassen von Pixeln
Darstellung vergrößert

12.4.2. Vergrößern von Bildern (Zooming)

Erfolgte die Abtastung eines digitalen Bildes ideal unter Einhaltung des Abtasttheorems (11.2), enthält es alle Information der Originalfunktion. Dann beschreibt das trigonometrische Interpolations-Polynom die Originalfunktion exakt. Für eine Vergrößerung des Bildes reicht es deshalb, die zusätzlichen Zwischenpixel durch trigonometrische Interpolation zu berechnen. Nicht immer liefert das optimale Ergebnisse. Wurde die "ideale Abtastung" erreicht, indem die Bandbegrenzung durch eine vorherige Filterung hergestellt wurde, sind dadurch Informationen verloren gegangen. Bei einem idealen Tiefpass beispielsweise wurden dadurch Ringing-Artefakte erzeugt, die im vergrößerten Bild sichtbar werden. Andere Interpolationsverfahren können diese Effekte vermindern. Im einfachsten Fall kann eine einfache Verdopplung der Pixel (Pixel-Replikation) ausreichende Resultate erzielen. Die Wahl des Verfahrens ist hier sehr subjektiv und vom Bildmaterial abhängig.

Teil VI.

Bestimmung geometrischer Transformationen

13. Bildtransformationen

Bilder werden aus den verschiedensten Gründen transformiert. Dies können geometrische Transformationen sein, aber auch Darstellungen in anderen Koordinatensystemen. Eine solche Transformation lässt sich wie folgt beschreiben: Sei $\mathbf{x}' = f(\mathbf{x})$ die geometrische Transformation, die auf das Bild angewendet werden soll. Dann muss das transformierte Pixel im Zielbild g' den gleichen Wert wie das originale Pixel im Ausgangsbild g haben.

$$g'(x') = g'(f(\mathbf{x})) = g(\mathbf{x}) \tag{13.1}$$

13.1. Globale geometrische Transformationen

- Translationen

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix} \quad \leftrightarrow \quad \mathbf{x}' = \mathbf{x} + \mathbf{a}_0 \tag{13.2}$$

  Bei einer Anwendung auf Funktionen bedeutet dies

$$g'(x+a) = g(x) \quad \text{oder} \quad g'_{n+k} = g_n$$

  Für eine explizite Darstellung von g' substituieren wir $x + a \to x$ bzw. $n + k \to n$:

$$g'(x) = g(x-a) \quad \text{oder} \quad g'_n = g_{n-k}$$

  Diese Operation haben wir bereits als Verschiebungsoperator S eingeführt (1.4):

$$g' = S_a g \quad \text{oder} \quad g' = S_k g$$

- Euklidische Transformationen

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{a}_0 \tag{13.3}$$

  $\mathbf{R}$ ist dabei eine orthogonale Matrix. Diese Transformationen bestehen also aus Translationen, Rotationen und eventuell Spiegelungen. Wenn man Spiegelungen ausschließt, dann ist $\mathbf{R}$ eine Rotationsmatrix, hat also die Form

$$\mathbf{R} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \tag{13.4}$$

Manchmal betrachten wir auch nur "reine" Rotationen, Rotationen um den Ursprung

$$\mathbf{x}' = \mathbf{R}\mathbf{x} \tag{13.5}$$

Wenn wir einen Punkt (x, y) in der Ebene als komplexe Zahl $z = x + iy$ auffassen, dann können wir reine Rotationen in der komplexen Ebene kompakt formulieren

$$z' = z \cdot e^{i\varphi} \tag{13.6}$$

- Scherungen
  Eine x-Scherung ist

$$\begin{aligned} x' &= x + s_x y \\ y' &= y \end{aligned} \tag{13.7}$$

  Eine y-Scherung ist

$$\begin{aligned} x' &= x \\ y' &= s_y x + y \end{aligned} \tag{13.8}$$

- Ähnlichkeitstransformationen
  Ähnlichkeitstransformationen enthalten zusätzlich zu den Euklidischen Transformationen die Möglichkeit einer Größenänderung (Skalierung)

$$\mathbf{x}' = s \cdot \mathbf{R}\mathbf{x} + \mathbf{a}_0 \quad \text{mit} \quad s > 0 \tag{13.9}$$

  Es kommt zusätzlich zu den Euklidischen Transformationen noch eine isotrope Skalierung s hinzu.

- Anisotrope Skalierung
  s in der Gleichung 13.9 der Ähnlichkeitstransformationen beschreibt eine isotrope Skalierung. Manchmal benötigen wir auch noch eine anisotrope Skalierung, bei der die Koordinaten unterschiedlich skaliert werden.

$$\begin{aligned} x' &= c \cdot x \\ y' &= d \cdot y \end{aligned} \tag{13.10}$$

- Affine Transformationen
  Alle bisher genannten Transformationen sind spezielle affine Transformationen. Affine Transformationen lassen sich allgemein formulieren als

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{a}_0 \tag{13.11}$$

  Dabei ist A eine beliebige 2x2-Matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{13.12}$$

Affine Transformationen lassen sich in der komplexen Ebene einfach darstellen. Sei $z = x + iy$, dann wird die affine Transformation beschrieben durch

$$z' = a \cdot z + b \cdot \overline{z} + c \tag{13.13}$$

Die (komplexen) Parameter a, b und c lassen sich aus der Matrix $\mathbf{A}$ und Vektor $\mathbf{a_0}$ berechnen

$$\begin{aligned} a &= \frac{1}{2}(a_{11} + a_{22}) + i\frac{1}{2}(a_{21} - a_{12}) \\ b &= \frac{1}{2}(a_{11} - a_{22}) + i\frac{1}{2}(a_{21} + a_{12}) \\ c &= a_{10} + ia_{20} \end{aligned}$$

- Projektive Transformationen

$$\begin{aligned} x' &= \frac{a_{11}x + a_{12}y + a_{10}}{a_{31}x + a_{32}y + a_{30}} \\ y' &= \frac{a_{21}x + a_{22}y + a_{20}}{a_{31}x + a_{32}y + a_{30}} \end{aligned} \tag{13.14}$$

13.2. Polar- und Log-Polar-Darstellungen

Abbildung 13.1.
Lena

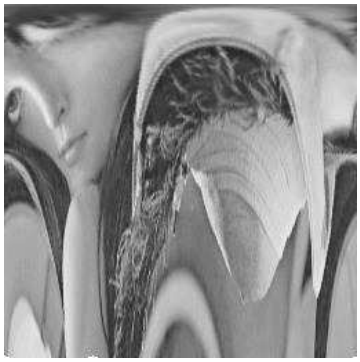


Abbildung 13.2.
Lena, Polar-Bild, $r \geq 1$

Abbildung 13.3.
Lena, Log-Polar-Bild, $r \geq 1$

Bei der *Polar-Darstellung* (13.2) werden die kartesischen Koordinaten (x, y) in Polarkoordinaten (r, φ) umgerechnet, wobei $x = r\cos\varphi$ und $y = r\sin\varphi$ gilt. r und φ dienen dann als Koordinaten im Zielbild.

$$f^{pol}(r, \varphi) = f(r\cos\varphi, r\sin\varphi)$$

Bei der *Log-Polar-Darstellung* (13.3) wird im Unterschied zur Polar-Darstellung der Radius logarithmisch dargestellt. Anstelle (r, φ) werden die Koordinaten $(\log r, \varphi)$ verwendet. Durch die Logarithmierung ist es nicht möglich, mit $r = 0$ zu beginnen, man fängt daher erst mit $r \geq 1$ an.

$$f^{logpol}(\log r, \varphi) = f(r \cos \varphi, r \sin \varphi)$$

Generell stehen für kleine r nur wenige Pixel mit Grauwerten zur Verfügung. Die entsprechenden Zeilen im Zielbild tragen deshalb nur wenig Information. Aus praktischen Gründen wird man deshalb oft nicht bei $r = 1$, sondern bei größeren Werten beginnen. In der Abbildung 13.3 wurde die Log-Polar-Darstellung mit $r = 1$ begonnen. Man sieht den auffälligen homogenen Bereich, der kaum eine Bildinformation trägt.

13.3. Praktische Durchführung von Transformationen

Wenn wir ein diskretes Bild g mit einer Transformation $\mathbf{f}(\mathbf{x})$ in ein diskretes Bild g' transformieren wollen, könnten wir zunächst wie in Gleichung 13.1 vorgehen: Wir nehmen alle Punkte des Originalbild $g(\mathbf{x})$, bestimmen den jeweiligen Zielpunkt $\mathbf{x}' = f(\mathbf{x})$ und tragen dort den Wert aus dem Originalbild ein.

Abbildung 13.4.
Lena

Abbildung 13.5.
Lena transformiert nach 13.1

Dabei fallen uns zunächst zwei Spezialfälle auf: Zu einigen Punkten des Originalbildes liegen die Zielpunkte außerhalb des Zielbildes. Dann können wir den Wert nicht eintragen, der Inhalt wird "aus dem Bild herausgeschoben". Andere Flächen im Zielbild bleiben leer, weil die entsprechenden Punkte im Originalbild nicht existieren. Beide Fälle sind normal und wegen des endlichen Definitionsbereiches der Bilder nicht vermeidbar.
Problematisch ist dagegen ein anderer Effekt: Auch in Bereichen, die im Prinzip innerhalb des transformierten Bildes liegen, werden Pixel des Zielbildes nicht beschrieben. Das resultiert daraus, dass wir nur die diskreten Punkte im Originalbild behandelt haben und diese

nicht alle Punkte im Zielbild füllen müssen. Ehe wir uns jetzt eine Methode überlegen, wie wir die fehlenden Punkte nachträglich füllen können, sollten wir unsere ganze Vorgehensweise überdenken. Können wir nicht eine Definition finden, wie die Pixel des Zielbildes zu belegen sind? Das ist nicht schwer: Wir substituieren in Gleichung 13.1 einfach x durch $f^{-1}(x)$:

$$g'(f(f^{-1}(\mathbf{x}))) = g'(\mathbf{x}) = g(f^{-1}(\mathbf{x})) \tag{13.15}$$

Wir berechnen jetzt das transformierte Bild, indem wir zu jedem Pixel $g'(\mathbf{x})$ die entsprechenden Koordinaten im Originalbild ermitteln und den dortigen Wert $g(f^{-1}(\mathbf{x}))$ ins Zielbild übernehmen.

Abbildung 13.6.
Lena

Abbildung 13.7.
Lena transformiert nach 13.15

Nun wird $f^{-1}(\mathbf{x})$ nicht immer ganzzahlige Koordinatenwerte aufweisen. Wir könnten die Koordinatenwerte einfach runden und das nächstgelegene Nachbarpixel erreichen. Eine bessere Variante ist, zwischen den benachbarten Pixeln zu interpolieren. Die häufigste Interpolationsvariante ist die *bilineare Interpolation* der vier benachbarten Pixelwerte.

14. Signalbasierte Transformations-Bestimmung

In vielen Anwendungen der Bildverarbeitung müssen zwischen Bildern (Funktionen) Transformationen bestimmt werden. Welche Transformationen benötigt werden, hängt natürlich von der konkreten Anwendung ab. Ein typisches Gebiet ist die **Bildregistrierung**, die zwei Bilder aneinander anpassen will, indem sie Transformationen eliminiert.
Die Transformationen können durch Punktkorrespondenzen berechnet werden. Die Punktkorrespondenzen basieren auf verschiedensten Merkmale und Methoden, ein Beispiel sind SIFT-Merkmale.
Transformationen können aber auch ohne Punktkorrespondenzen ermittelt werden. In den folgenden Abschnitten sollen solche signalbasierten Methoden betrachtet werden werden. Wenn wir allgemeine Transformationen betrachten, müssen wir eigentlich das analoge Modell zugrunde legen. Im diskreten gibt es keine Rotationen, Skalierungen und so weiter. Das diskrete Modell, indem wir letztendlich rechnen, ist eine Näherung, die aus praktischen Gründen notwendig ist. Translationen dürfen wir im diskreten Modell mit ganzzahlige Verschiebungen zulassen. Wenn ein Bild nicht unterabgetastet ist, dann kann die analoge Funktion durch Interpolation vollständig rekonstruiert werden und die Registrierung mittels Interpolation ist korrekt. Das bedeutet, dass man dann auch Verschiebungen im Subpixelbereich ermitteln kann.

14.1. Translation

Die wichtigste Transformation ist wohl die Verschiebung oder Translation zweier Bilder. Viele andere Transformationsbestimmungen lassen sich auf die Bestimmung von Translationen zurückgeführt.

14.1.1. Anwendung des Verschiebungstheorems

Eine naheliegende Methode zur signalbasierten Translationsbestimmung ist die Anwendung des Verschiebungstheorems (Abschnitt 7.8).
Wir schreiben dies einmal auf für diskrete zweidimensionale Bilder der Dimension M, N:

$$\alpha_{k,l}(S_{m,n}f) \;=\; \alpha_{k,l}(f) \cdot e^{-2\pi i(m\frac{k}{M}+l\frac{n}{N})}$$

Da sich die Amplitude nicht ändert, können wir die Translation aus obiger Beziehung allein über die Phase ausrechnen. Dazu benötigen wir nur zwei Koeffizienten, um m und n zu

berechnen. Wir bezeichnen mit $\phi_{k,l}(f)$ die Phase von $\alpha_{k,l}(f)$. Dabei müssen wir beachten, dass die Phasen wegen der Periodizität mehrdeutig sind. Zur Vereinfachung der Rechnung verwenden wir $\phi_{k,0}$ und $\phi_{0,l}$:

$$\begin{aligned} m &= \frac{(\phi_{k,0}(f) - \phi_{k,0}(S_{m,n}f) \cdot M}{2\pi} + j_1 \cdot \frac{M}{k} \\ n &= \frac{(\phi_{0,l}(f) - \phi_{0,l}(S_{m,n}f) \cdot N}{2\pi} + j_2 \cdot \frac{N}{l} \end{aligned}$$

Die Notation der Periode ist sehr wichtig. Mit $k = 1$ oder $l = 1$ ist die Mehrdeutigkeit die Dimension des Bildes und lässt sich mit etwas a-priori-Wissen leicht eliminieren. Ähnliches gilt noch bei niedrigen Werten für k und l. Allerdings werden die Phasen für kleine k und l am meisten durch wrap-around-Effekte beeinflusst und sind deshalb fehlerhafter. Eine geignete Kombination verschiedener Werte erlaubt, auch bei höheren k und l die Mehrdeutigkeit aufzulösen.

14.1.2. Kreuzkorrelation

Die Kreuzkorrelation ist ein Synonym für die Korrelation (2.4):

$$(f \circ g)_n = \sum_i f_{n+i}\overline{g_i} \qquad (f \circ g)_{m,n} = \sum_i \sum_j f_{m+i,n+j}\overline{g_{i,j}}$$

$$(f \circ g)(t) = \int_B f(t+\xi)\overline{g(\xi)}d\xi \quad (f \circ g)(t_1,t_2) = \iint_B f(t_1+\xi,t_2+\eta)\overline{g(\xi,\eta)}d\xi d\eta$$

Wenden wir die Kreuzkorrelation auf zwei gegeneinander verschobene Bilder g und $S_m g$ an, so erhalten wir:

$$S_m g \circ g = (S_m g) * g^R = S_m(g * g^R) = S_m(g \circ g)$$

Das Ergebnis ist also die um m verschobene Autokorrelation. Da die Autokorrelation ihr Maximum immer bei 0 hat, liefert uns die Lage des Maximums der Kreuzkorrelation die Verschiebung.

14.1.3. Translation als LSI-Operator

Translationen oder Verschiebungen von Funktionen haben wir mit dem Verschiebungsoperator beschrieben. Zu den Eigenschaften des Verschiebungsoperators gehört, dass es sich um einen LSI-Operator handelt, den wir also durch eine Faltung beschreiben können:

$$f' = S_m f = S_m(f * \delta) = f * S_m\delta$$

Wir können nun die bekannten Methoden anwenden, um bei gegebenen Bildern f und f' die Faltungsmaske $S_m\delta$ zu ermitteln. Im Idealfall ist das eine Funktion, die genau an einer Stelle, der Verschiebung m, den Wert 1 aufweist, sonst den Wert 0. Damit wäre

die Verschiebung aus der Maske leicht zu ermitteln. Bei realen Bildern mit Störungen wird das Ergebnis kein idealer verschobener Einheitsimpuls sein, sondern eine Funktion, die hoffentlich ein ausgeprägtes deutliches Maximum (**Peak**) besitzt. Die Schätzung der Verschiebung ist dann durch Ermittlung des Maximums weiter möglich.

14.1.4. Inverse Faltung

In der einfachsten Form berechnen wir die Verschiebungsmaske mittels Faltungsinvertierung.

$$f' = S_n f = S_n(f * \delta) = f * S_n\delta \quad \rightarrow \quad \alpha_k(S_n\delta) = C \cdot \frac{\alpha_k(f')}{\alpha_k(f)} \tag{14.1}$$

Wir haben hier die Verschiebung als zyklisch betrachtet. Bei praktischen Aufgaben ist die Verschiebung fast nie zyklisch. Bildteile werden durch die Verschiebung aus dem Bild geschoben und neue Bildteile erscheinen auf der anderen Seite. Wir müssen dies als Fehler zusätzlich zum Rauschen betrachten. Wie wir in Abschnitt (9) gezeigt haben, werden Rauschen und andere Fehler bei der inversen Filterung verstärkt und die Methode wird oft praktisch unbrauchbar. Besser ist es deshalb, die Berechnung des verschobenen Einheitsimpulses als Restaurationsproblem aufzufassen.

14.1.5. Phasenkorrelation

Die Phasenkorrelation ist eine recht robuste Methode zur Translationsbestimmung. Sie wurde von Kuglin und Hines [23] bereits 1975 eingeführt. Wir nehmen wieder an $f' = S_n f$ sei die Verschiebung von f. Vom Verschiebungstheorem $\alpha_k(f') = \alpha_k(f)e^{-2\pi i n\frac{k}{N}}$ wissen wir, dass das Amplitudenspektrum invariant ist, also ist $|\alpha_k(f')| = |\alpha_k(f)|$. Wir beschreiben wieder die Translation als Faltung:

$$f' = S_n f = S_n(f * \delta) = f * S_n\delta \quad \rightarrow \quad \alpha_k(S_n\delta) = C \cdot \frac{\alpha_k(f')}{\alpha_k(f)}$$

Nun erweitern wir einfach Zähler und Nenner und nutzen die Invarianz des Amplitudenspektrums:

$$\alpha_k(S_n\delta) = C \cdot \frac{\alpha_k(f')}{\alpha_k(f)} = C \cdot \frac{\alpha_k(f')\overline{\alpha_k(f)}}{\alpha_k(f)\overline{\alpha_k(f)}} = C \cdot \frac{\alpha_k(f')\overline{\alpha_k(f)}}{|\alpha_k(f)|^2} = C \cdot \frac{\alpha_k(f')\overline{\alpha_k(f)}}{|\alpha_k(f')||\alpha_k(f)|}$$

Damit erhalten wir für die Phasenkorrelation

$$\alpha_k(S_n\delta) = C \cdot \frac{\alpha_k(f')}{|\alpha_k(f')|} \cdot \frac{\overline{\alpha_k(f)}}{|\alpha_k(f)|} \tag{14.2}$$

Auf der linken Seite steht das Spektrum des verschobenen Einheitsimpulses. Auf der rechten Seite sehen wir zunächst zwei Brüche: Es ist eine Normierung der Fourierkoeffizienten beider Bilder auf den Betrag 1 - das entspricht dem Whitening beider Ausgangsbilder

(7.15), wonach mit dem Produkt die Kreuzkorrelation berechnet wird. Alternativ könnt man es auch als ein Whitening der Kreuzkorrelation auffassen. Damit abstrahieren wir von der Amplitude der Kreuzkorrelation und nutzen nur noch die Phase - daher auch der Name Phasenkorrelation.

Abbildung 14.1.
Originalbild

Abbildung 14.2.
Verschobenes Bild

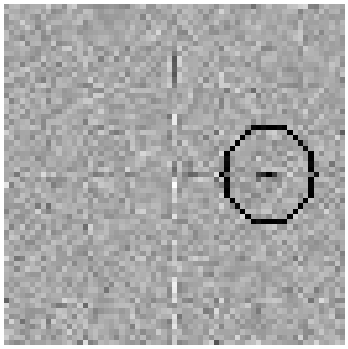


Abbildung 14.3.
Phasenkorrelation

In den Abbildungen 14.1, 14.2 und 14.3 ist links das Originalbild, in der Mitte das verschobene Bild und rechts das Ergebnis der Phasenkorrelation zu sehen. Der deutlich zu sehende Peak, der die Verschiebung bestimmt, wurde durch einen Kreis markiert. Trotz schlechter Bildqualität und Störungen kann der Peak sicher gefunden werden.

14.1.6. SDR-Methode

Die Methode **S**hift **D**etection by **R**estoration fasst die Bestimmung des verschobenen Einheitsimpulses als Restaurationsproblem auf:

$$f' = f * S_n\delta + N \tag{14.3}$$

Das Rauschen N beschreibt neben dem Pixelrauschen auch alle anderen Störungen und Fehler. Ein solcher Fehler ist unsere Annahme einer zyklischen Verschiebung. Dieser Fehler ist offensichtlich signal-abhängig und vorher schwer abzuschätzen. Das Wiener-Helstrom-Optimal-Filter (Abschnitt 10.3) ist damit nicht einsetzbar. Die Restauration unter Zwang (Abschnitt 10.2) benötigt nur wenig a-priori-Wissen: eine Schranke für das Rauschen. Bei der Wahl des Operators L entscheiden wir uns für die δ-Funktion und kommen zur Formel (10.3), die wir noch einmal für (14.3) aufschreiben:

$$\alpha_k(S_n\delta) = C \cdot \frac{\alpha_k(f') \cdot \overline{\alpha_k(f)}}{|\alpha_k(f)|^2 + \beta} \tag{14.4}$$

Die Größe β beschreibt die Stärke des Rauschens und muss empirisch festgelegt werden. Sie kann meist in einem weiten Bereich gewählt werden.
Für $\beta \to 0$ geht (14.4) in die inverse Faltung (Abschnitt 14.1.4) über. Für $\beta \to \infty$ wiederum geht (14.4) in die Kreuzkorrelation über (Abschnitt 14.1.2). Daher ist die SDR-Methode

sowohl auffassbar als regularisierte inverse Faltung, als auch als regularisierte Kreuzkorrelation.
Grundsätzlich kann man den verschobenen Einheitsimpuls aus (14.3) auch mit den anderen Restaurationsmethoden berechnen, die im Abschnitt 10 vorgestellt wurden.

14.1.7. Cepstrum-Methode

Es sei ein Bild gegeben, in welchem ein Bild und ein verschobenes Bild überlagert sind. Dieses Bild könnte durch eine Mehrfachbelichtung entstanden sein. Es stehe jetzt die Aufgabe, aus diesem Bild g die Verschiebung der überlagerten Bilder untereinander zu bestimmen.

$$g = f + f' = f + S_n f = f * \delta + S_n f * \delta = f * (\delta + S_n \delta)$$

Wäre das Originalbild f bekannt, ließe sich leicht der Verschiebungsterm $\delta + S_n\delta$ berechnen. Offensichtlich lässt sich aus dem gegebenen Bild g zu jeder Verschiebung ein Originalbild f berechnen. Wir können aber hoffen, dass typische Eigenschaften normaler Bilder die Trennung von Originalbild f und Verschiebungsterm $\delta + S_n\delta$ ermöglichen. Durch Anwendung des Faltungstheorems wird die Faltung in ein Produkt überführt, durch Logarithmieren erhalten wir eine Summe. Das erinnert an die Definition des Cepstrum in Abschnitt 7.18. Für die folgende Herleitung wollen wir wieder den eindimensionalen Fall im Modell D1(N) betrachten.

$$\alpha_k(g) = \alpha_k(f * (\delta + S_n\delta)) = \sqrt{N}\alpha_k(f)\alpha_k((\delta \ + \ S_n\delta))$$

Nun ist $\alpha_k(\delta) = \frac{1}{\sqrt{N}}$ und $\alpha_k(S_n\delta) = \frac{1}{\sqrt{N}} e^{-2\pi i n \frac{k}{N}}$.

$$\alpha_k(g) = \alpha_k(f) \cdot (1 + e^{-2\pi i n \frac{k}{N}})$$

Das Leistungsspektrum entsteht durch Multiplikation mit dem konjugiert komplexen:

$$|\alpha_k(g)|^2 = |\alpha_k(f)|^2 (1 + e^{-2\pi i n \frac{k}{N}})(1 + e^{+2\pi i n \frac{k}{N}})$$

Nach Logarithmieren erhalten wir:

$$\log |\alpha_k(g)|^2 = \log |\alpha_k(f)|^2 + \log(1 + e^{-2\pi i n \frac{k}{N}}) + \log(1 + e^{+2\pi i n \frac{k}{N}})$$

Zur Berechnung der Terme $\log(e^{\pm 2\pi i n \frac{k}{N}})$ benutzen wir die Reihenentwicklung:

$$\log(1 + x) \ = \ \sum_{m=1}^{\infty} (-1)^{m+1} \frac{x^m}{m}$$

Diese konvergiert für $|x| \leq 1$.

$$\begin{aligned} \log(1 + e^{\pm 2\pi i n \frac{k}{N}}) &= \sum_{m=1}^{\infty} (-1)^{m+1} \frac{e^{\pm 2\pi i n m \frac{k}{N}}}{m} \\ &= \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sqrt{N} \alpha_k(S_{\pm n \cdot m}\delta) \end{aligned}$$

Es ist ein sehr schönes Ergebnis, dass wir hier die Fourierkoeffizienten der verschobenen Einheitsimpulse wieder finden, denn im nächsten Schritt zum Cepstrum führen wir die inverse Fouriertransformation aus:

$$\begin{aligned}\alpha^{-1}(\log|\alpha_k(g)|^2) &= \alpha^{-1}(\log|\alpha_k(f)|^2) + \alpha^{-1}(\sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}\alpha_k(S_{nm}\delta) + \sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}\alpha_k(S_{-nm}\delta)) \\ &= \alpha^{-1}(\log|\alpha_k(f)|^2) + \sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}(S_{+n\cdot m}\delta) + \sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}(S_{-n\cdot m}\delta)\end{aligned}$$

Noch handelt es sich nicht um das power cepstrum, da wir beide Seiten noch quadrieren müssten. Da dies nur die Höhe der einzelnen Werte, nicht aber deren Ordnung ändern würde, verzichten wir darauf und nennen das Ergebnis ,,half power cepstrum" hpc.

$$hpc(g) = hpc(f) + \sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}(S_{+n\cdot m}\delta) + \sum_{m=1}^{\infty}\frac{(-1)^{m+1}}{m}\sqrt{N}(S_{-n\cdot m}\delta)$$

Das half power cepstrum des doppelbelichteten Bildes g setzt sich zusammen aus dem half power cepstrum des Original-Bildes f und Folgen von verschobenen Einheitsimpulsen mit wechselndem Vorzeichen und fallender Stärke. Die Verschiebungen sind Vielfache der gesuchten Verschiebung der doppelt belichteten Bilder.
Die verschobenen Einheitsimpulse der gesuchten Verschiebung sind eingebettet in das cepstrum des Ausgangsbildes f. Das ist aber meist kein Problem, da sich diese aus dem homogeneren Cepstrum des Bildes herausheben.

14.2. Translationen und Subpixel-Genauigkeit

Bei vielen Methoden zur Translationsbestimmung haben wir zum Schluss das Spektrum zurücktransformiert und das Maximum bestimmt. Die Koordinaten des Maximums nehmen wir als Koordinaten der Verschiebung. Da wir dies aber im Quadratgitter der Pixel tun, kann das Maximum nur ganzzahlig bestimmt werden. Häufig wird dann in der Praxis an die Punkte in der Nähe des Maximums eine Funktion gefittet, zum Beispiel eine Parabel, und anschließend das Maximum dieser Funktion bestimmt. Dies liefert oft gute Ergebnisse, ist aber nur ein heuristisch begründetes Herangehen.
Ein signaltheoretisch begründetes Vorgehen beruht auf dem Abtasttheorem: Wenn die zwei Originalbilder bandbegrenzt sind, so dass wir aus den Pixeln das Original rekonstruieren können, dann ist das Ergebnisbild, zum Beispiel die Kreuzkorrelation, auch bandbegrenzt

und kann auch vollständig rekonstruiert werden. Wir müssten dann das Maximum des trigonometrischen Interpolationspolynoms bestimmen und erhalten die Koordinaten mit Subpixel-Genauigkeit.
Eine in der Handhabung einfachere Methode ist das zeropadding: Bevor wir das Ergebnisspektrum zurücktransformieren, vergrößern wir es um den Faktor f und füllen mit Nullen auf - schließlich wissen wir aus der Bandbegrenzung, dass diese Fourierkoeffizienten Null sein müssen. Das Bild nach der Rücktransformation stellt ein höherabgetastetes Bild des analogen Ergebnisses dar. Das Maximum hat immer noch ganzzahlige Koordinaten, diese repräsentieren dann aber $1/f$ der Ausgangspixel.
In [13] wurden verschiedene Interpolations-Methoden zur subpixelgenauen Registrierung verglichen. Die Fourierbasierte trigonometrische Interpolation lieferte mit die besten Ergebnisse, allerdings steht dieser der hohe Rechenaufwand entgegen.

14.3. Reine Rotationen

Liegen nur Rotationen um den Ursprung vor, dann gibt es ein besonders einfaches signalorientiertes Verfahren: Wir führen durch Polar-Darstellung der Bilder (13.2) die Rotation auf eine Verschiebung zurück und bestimmen diese mit den bekannten Verfahren.
Um diese Translation und damit Rotation effizienter zu bestimmen, bilden wir aus den Polarbildern die $1D$-Funktion

$$h(\alpha) = \int_{\alpha-\Delta\alpha}^{\alpha+\Delta\alpha} \int_{0}^{\infty} f^{pol}(r,\varphi)\ dr\ d\varphi$$

Jetzt müssen wir nur noch die Verschiebung dieser "$1D$-Bilder" berechnen.
Für die diskreten endlichen Bilder müssen wir ein paar zusätzliche Überlegungen anstellen. Der Rand ragt nach der Transformation unterschiedlich weit in das Bild. Deshalb sollten wir den Radius nur soweit ausdehnen, das der Rand an keiner Stelle hineinragt. Zum anderen erhalten wir durch die Diskretisierung je nach φ unterschiedliche Pixelanzahlen. Besser ist es deshalb, statt des Integrals den Mittelwert in den einzelnen Sektoren zu berechnen.

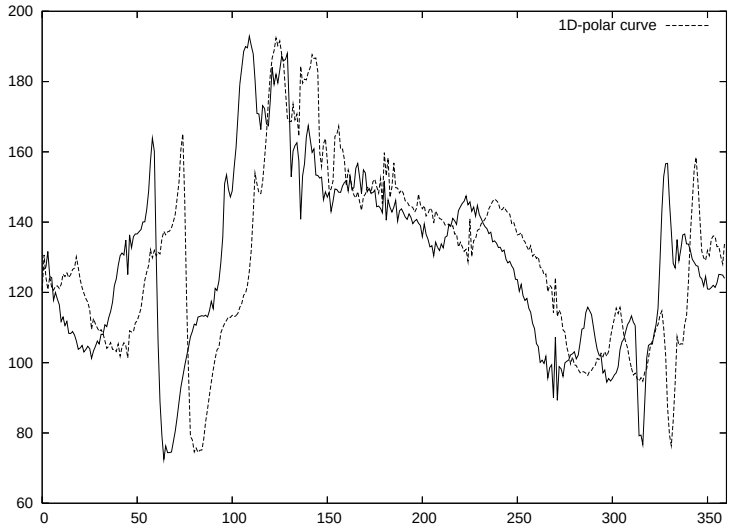


Abbildung 14.4.
Verschobene 1D-Polarfunktionen

Im Beispiel wurden das Lena-Bild aus Abb.13.1 und das um 11 Grad rotierte Lena-Bild in diese 1D-Polarform überführt. In der Abb. 14.4 sieht man deutlich die Verschiebung der beiden Kurven zueinander. In den meisten praktischen Anwendungen reicht diese 1D-Methode aus. Man muss sich aber darüber im klaren sein, dass durch Überführung des 2D-Polarbild in ein 1D-Polarbild Information verloren geht. Es kann also Bilder geben, bei denen die Translation auf den 2D-Polarbildern bestimmt werden muss, weil dies auf den 1D-Polarbildern nicht mehr funktioniert.

14.4. Euklidische Transformationen

Eine sehr häufig vorkommende Transformation ist die Rotation verbunden mit Translationen. Um mit den bekannten Mitteln das Problem der Bestimmung dieser Transformation zu lösen, gehen wir wie folgt vor:

- Wir eliminieren zunächst die Translation, indem wir von beiden Bildern das Leistungsspektrum bilden. Dieses ist verschiebungsinvariant, enthält aber die gleiche “reine” Rotation.
- Wir bestimmen die Rotation, indem wir sie mittels Polardarstellung in eine Translation überführen, wie wir im Abschnitt 14.3 gezeigt haben.
- Wir korrigieren die Rotation und haben zwei Bilder mit einer reinen Translation.
- Eines der bekannten Verfahren zur Translationsbestimmung benutzen wir zur Bestimmung der Translation.

14.5. Ähnlichkeitstransformationen

Kommen zu den Euklidischen Transformationen noch isotrope Skalierungen hinzu, sprechen wir von Ähnlichkeitstransformationen. Die bestehen somit in der Ebene aus vier zu bestimmenden Parametern, der Translation (2 Parameter), der Rotation (1 Parameter) und der isotropen Skalierung (1 Parameter). Nun nutzen wir die gleiche Idee wie im Abschnitt 14.4 zur Eliminierung der Translation. Wir bilden von beiden Bildern die Leistungs- oder Amplitudenspektren, welche Translations-invariant sind. Nun wollen wir wieder die Bestimmung der restlichen Parameter (Rotation und Skalierung) auf die Bestimmung einer $2D$-Translation zurückführen. Dazu nutzen wir die Log-Polar-Darstellung (13.2), die die Rotation in eine Translation in φ-Richtung und die Skalierung in eine Translation in $\log(r)$-Richtung überführt. Nach der Überführung der Spektren in die Log-Polar-Darstellung bestimmen wir über die $2D$-Translation die Rotation und Skalierung. Anschließend transformieren wir die beiden Originalbilder mit diesen beiden Parametern, und bestimmen aus diesem Bildpaar die verbleibende Translation.

14.6. Affine Transformationen

Die signalbasierte Bestimmung affiner Transformationen (13.11) ohne explizite Referenzpunkte ist deutlich schwieriger. Die affine Transformation

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{a}_0$$

lässt sich in einen linearen Anteil $\mathbf{A}\mathbf{x}$ und eine Translation $\mathbf{a}_0$ zerlegen. Die Benutzung des Amplitudenspektrum sorgt wieder für Verschiebungs-Invarianz und reduziert die Aufgabe auf die Bestimmung der Matrix A aus dem Spektrum. Dazu werden in der Regel affin covariante Merkmale benutzt, siehe zum Beispiel [22], [26]. Anschließend wird ein Bild mit dem linearen Anteil der affinen Transformation transformiert und dann die Translation bestimmt.

14.7. Translations-ähnliche Transformationen

Oft sind die Transformationen zwischen zwei Bildern recht einfach. Nimmt man mit einer Kamera aus der Hand ein Motiv zwei mal nacheinander auf, so werden die Bilder gegeneinander “etwas mehr als verschoben” sein. Eine Translation ist hier als Modell ungenügend, eine affine Transformation beschreibt die Situation besser. Um diese affine Transformation zu bestimmen, kann die Ähnlichkeit mit der Translation hilfreich sein: Betrachtet man einen Ausschnitt/Block des Bildes, kann man dort lokal die Transformation durch eine Translation nähern.
Diese Idee wird beispielsweise beim Blockmatching genutzt: Nacheinander betrachtet man jeweils einen Teilblock des ersten Bildes und sucht im zweiten Bild nach dem ähnlichsten Block (Abbildung 14.5).

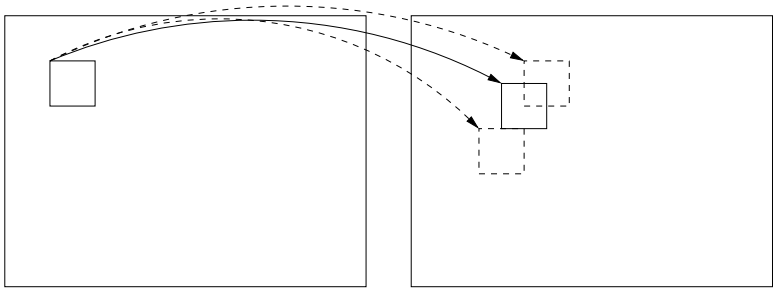


Abbildung 14.5.
Suche des ähnlichsten Blockes beim Blockmatching

Von den ermittelten Blockpaaren kann man zum Beispiel die Mittelpunkte als Referenzpunkte betrachten und mittels dieser Referenzpunktpaare die affine Transformation bestimmen (Abbildung 14.6).

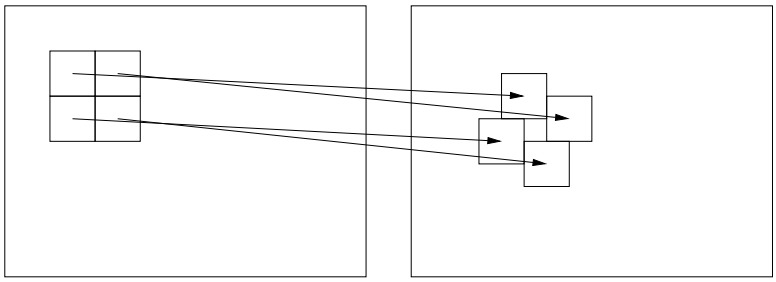


Abbildung 14.6.
Korrespondenzen aus der Blockposition

Bei der Suche nach dem ähnlichsten Block wird man das vorhandene a-priori-Wissen einsetzen, um den Suchbereich einzuschränken. Ein mögliches Ähnlichkeitsmaß ist die Differenz der Grauwerte in den Pixeln. Der Korrelationskoeffizient als Ähnlichkeitsmaß hat den Vorteil, von Beleuchtungsschwankungen unabhängig zu sein.
Mit Hilfe unserer signalbasierten Methoden zur Verschiebungsdetektion können wir auch ohne Suche einen korrespondierenden Block bestimmen. Überlappen sich die Blöcke in den beiden Bildern ausreichend, so kann die Verschiebung zum Beispiel mit SDR (Abschnitt 14.1.6) bestimmt werden (Abbildung 14.7).

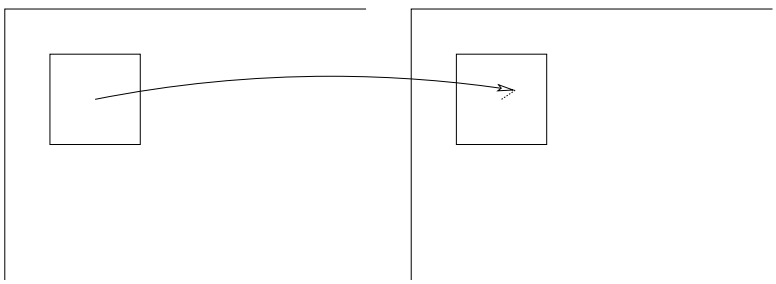


Abbildung 14.7.
Ermittlung der Blockverschiebung mittels SDR

Der Blockmittelpunkt im ersten Bild und der verschobene Punkt im zweiten Bild bilden nun ein Referenzpunktpaar. Aus allen Referenzpunktpaaren lässt sich wieder die affine Transformation berechnen (Abbildung 14.8).

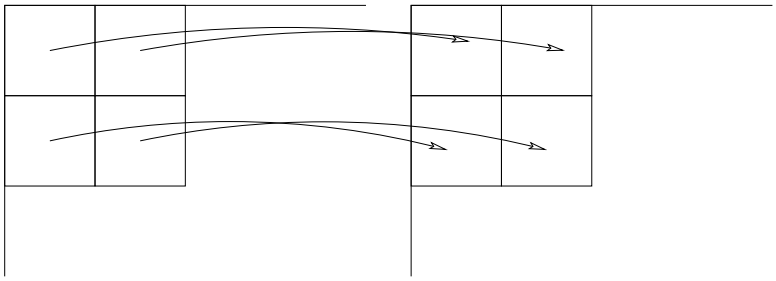


Abbildung 14.8.
Korrespondenzen aus den Verschiebungen

Beispiel Fundusbilder

In der Augenheilkunde werden Bilder des Augenhintergrundes (fundus oculi) erstellt, wobei nach Spritzen eines Kontrastmittels in Aufnahmen im Abstand einiger Minuten das Einströmen des Blutes mit Kontrastmittel beobachtet werden kann. Die Aufnahmen kommen wegen zwangsläufiger Bewegungen des Augapfels nicht zur Deckung. Eine entsprechende Anpassung (Registrierung) ermöglicht eine bessere Beurteilung der Bilder. Die beiden Beispielbilder in Abbildung 14.9 und 14.10 zeigen die starken Veränderungen, die Aufhellung durch das Kontrastmittel, die die Registrierung erschweren.

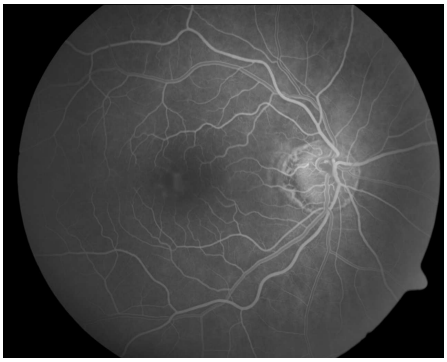


Abbildung 14.9.
fundus oculi - Referenzbild 1

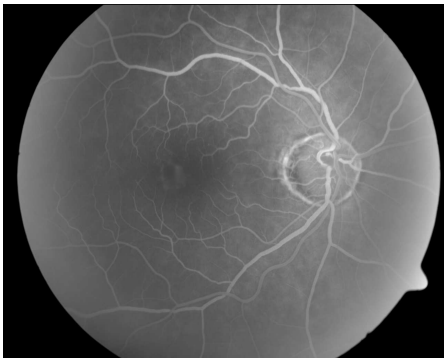


Abbildung 14.10.
fundus oculi - Referenzbild 1

Die Abbildung 14.11 verdeutlicht eine Stufe der Ausrichtung: nach dem Ausgleich der globalen Verschiebung verbleiben Differenzen im Bild rechts oben, die auf Bewegungen zurückzuführen sind, die über eine Verschiebung hinausgehen. Das Gesamtbild wird nun in 16 Blöcken unterteilt und mittels SDR die lokale Verscheibung bestimmt. Das Bild links unten ist das Ergebnisbild von SDR, das mittlere Bild zeigt die (vergrößerten) Verschiebungsvektoren. Nachdem die affine Transformation daraus ermittelt wurde, kann eine

entsprechende Ausrichtung der Bilder erfolgen. Die Differenz im Bild rechts unten zeigt die Veränderungen durch das Strömen des Kontrastmittels.

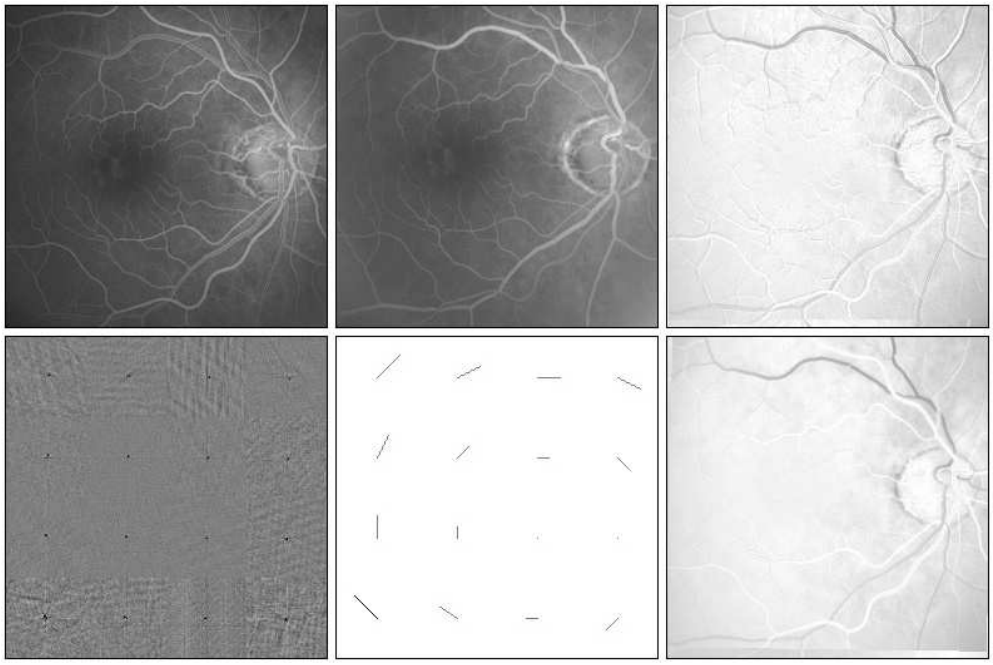


Abbildung 14.11.
Registrierungsschritt

Teil VII.

Konturen und Fourierdeskriptoren

15. Konturen als komplexwertige Funktion

15.1. Konturen als komplexwertige Funktion

In der Bildverarbeitung werden Objekte und Regionen oft mittels der Kontur beschrieben. Die Kontur ist die Randkurve, die das Objekt beziehungsweise die Region umschließt. Man kann zur Darstellung eine parameterische Beschreibung wählen, etwa $x(t), y(t)$ mit $t \in [0, T]$. t ist dabei irgendein Parameter. In der diskreten Darstellung können wir die Konturpunkte als Paare x_k, y_k mit $k = 0, ..., N-1$ aufzählen. Interpretieren wir beide Darstellungen in der komplexen Ebene, stellt jeder Konturpunkt eine komplexe Zahl $z = x + iy$ dar. Wir schreiben

$$\begin{aligned} z(t) &= x(t) + iy(t) \quad \text{mit} \quad t \in [0, T] \\ z_k &= x_k + iy_k \quad \text{mit} \quad k = 0, ..., N-1 \end{aligned} \tag{15.1}$$

Damit haben wir nun eine analoge oder diskrete Funktion mit einer unabhängigen Variablen erhalten. Die Funktionswerte sind komplex. Diese Funktionen sind periodisch: Wenn wir einmal um das Objekt herumgelaufen sind, erreichen wir erneut den Startpunkt und die Funktion wiederholt sich.

15.2. Trigonometrische Interpolation von Konturen

Aus diskreten Bildern bestimmte Konturen sind zunächst typischerweise eine diskrete Folge von Punkten oder bei Beschreibung in der komplexen Ebene eine diskrete komplexwertige Funktion:

$$z_k = x_k + iy_k \quad \text{mit} \quad k = 0, ..., N-1$$

Die trigonometrische Interpolation (Abschnitt 11.1) können wir auch hier anwenden, um zu einer analogen Funktion zu kommen:

$$z_{k_0}(t) = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k(z) e^{2\pi i k \frac{t}{N}} \tag{15.2}$$

In Abhängigkeit vom gewählten k_0 variieren die Interpolationsfunktionen stark (Abbildungen 15.1, 15.2 und 15.3). “Vernünftig” erscheinen glatte Funktionen, die entstehen, wenn

die genutzten Frequenzbeträge möglichst klein sind. Wir wählen $k_0 = -N/2$, also

$$z_{-\frac{N}{2}}(t) = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} \alpha_k(z) e^{2\pi i k \frac{t}{N}} \tag{15.3}$$

und damit das glatteste Polynom.

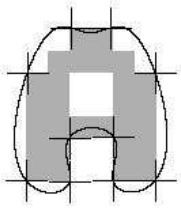


Abbildung 15.1.
Interpolation, $N = 10$,
$k_0 = -5$

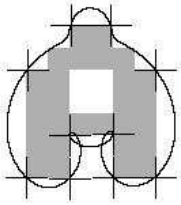


Abbildung 15.2.
Interpolation, $N = 10$,
$k_0 = -2$

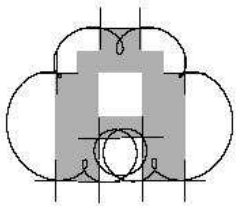


Abbildung 15.3.
Interpolation, $N = 10$,
$k_0 = 0$

Für Darstellungen und weitere numerische Behandlung der Konturen benötigt man wieder eine diskrete Darstellung, die aber deutlich mehr Stützstellen $M >> N$ aufweist. Die interpolierten Konturen kann man leicht erzeugen, indem man M Punkte mit $M >> N$ erzeugt und einträgt. Dazu unterteilen wir N in $t_m = m\frac{N}{M}$ und setzen in (15.3) ein und erhalten

$$z_m^* = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \alpha_k(z) e^{2\pi i k \frac{m}{M}}$$

15.3. Trigonometrische Approximation von Konturen

Im folgenden gehen wir vom trigonometrischen Interpolationspolynom der diskreten Konturpunkte aus, und zwar mit $k_0 = -N/2$, demzufolge

$$z(t) = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} \alpha_k(z) e^{2\pi i k \frac{t}{N}}$$

Wenn wir nun sukzessive hohe Frequenzen weglassen, dann erhalten wir kein Interpolationspolynom mehr, sondern die Kontur wird durch das Polynom nur noch approximiert. Diese trigonometrische Approximation ist im Grunde genommen nichts weiter als die An-

wendung eines Tiefpasses

$$z_P^{\circ}(t) = \frac{1}{\sqrt{N}} \sum_{k=-P}^{P} \alpha_k(z) e^{2\pi i k \frac{t}{N}} \quad \text{mit} \quad 0 < P < \frac{N}{2}$$

Je kleiner P wird, umso "glatter" werden die Konturen und umso größer werden die Abweichungen (Abbildungen 15.4, 15.5 und 15.6).

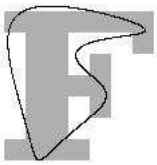


Abbildung 15.4.
Approximation P=3

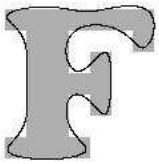


Abbildung 15.5.
Approximation P=10

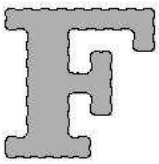


Abbildung 15.6.
Approximation P=50

15.4. Ellipsenfitting von Konturen

Für $P = 0$ ist das Ergebnis der trigonometrischen Approximation eine konstante Funktion und somit uninteressant. Ist $P = 1$ bleiben noch 3 Terme übrig:

$$z_1(t) = \frac{1}{\sqrt{N}} \sum_{k=-1}^{+1} \alpha_k(z) e^{2\pi i k \frac{t}{N}} = \frac{1}{\sqrt{N}} \left[\alpha_0 + \alpha_{-1} e^{-2\pi i \frac{t}{T}} + \alpha_1 e^{2\pi i \frac{t}{T}} \right] \tag{15.4}$$

Wenn wir die komplexe Schreibweise aufspalten und mit den Koordinaten x und y darstellen, erhalten wir die Parameterdarstellung einer geschlossenen Kurve:

$$x(t) = \frac{1}{\sqrt{N}} \left[Re(\alpha_0) + Re(\alpha_1 + \alpha_{-1}) \cdot \cos \frac{2\pi t}{T} + Im(\alpha_{-1} - \alpha_1) \cdot \sin \frac{2\pi t}{T} \right]$$

$$y(t) = \frac{1}{\sqrt{N}} \left[Im(\alpha_0) + Re(\alpha_1 - \alpha_{-1}) \cdot \sin \frac{2\pi t}{T} + Im(\alpha_{-1} + \alpha_1) \cdot \cos \frac{2\pi t}{T} \right]$$

Das ist eine bekannte parametrische Form der Darstellung einer **Ellipse**. Damit ist das Polynom $z_1(t)$ stets eine Ellipse und kann zum **Ellipsenfitting** von Konturen benutzt werden (Abbildungen 15.7,15.8 und 15.9).

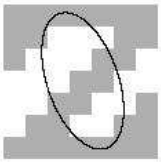


Abbildung 15.7.
$e_{form} = 0.656$

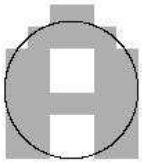


Abbildung 15.8.
$e_{form} = 0.076$

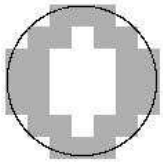


Abbildung 15.9.
$e_{form} = 0.006$

Es ist auch relativ leicht, ein Maß für die Abweichung der Kontur von der Ellipse anzugeben: Der Betrag der weggelassenen Fourierkoeffizienten ist ein Fehlermaß. Wir summieren die (quadrierten) Beträge dieser Fourierkoeffizienten und normieren bezüglich der beiden ersten Fourierkoeffizienten:

$$e_{form} = \frac{\sum\limits_{|k|>1} |\alpha_k(z)|^2}{|\alpha_{-1}(z)|^2 + |\alpha_1(z)|^2}$$

Diese Maße sind in den Abbildungen 15.7, 15.8 und 15.9 eingetragen und entsprechen auch unserer Vorstellung von der Abweichung von der Ellipsenform.

In [39] wird die Kovarianz dieses Ellipsenfittings bezüglich affiner Transformationen gezeigt: Wird eine Kontur affin transformiert, dann ist die gefittete Ellipse ebenso affin transformiert bezüglich der gefitteten Ellipse der Ausgangskontur. Dabei müssen natürlich die Stützstellen der transformierten Funktion exakt den transformierten Stützstellen der Ausgangsfunktion entsprechen. Das wird in der Praxis bei Konturen, die aus Bildern gewonnen werden, selten exakt der Fall sein. Deshalb muss man mit Abweichungen rechnen und zum Beipiel eine Reparametrisierung in Betracht ziehen.
Dies erklärt, warum das Ellipsenfitting nur dann plausible Ergebnisse liefert, wenn die vollständige Kontur gegeben ist. Ein Auschnitt der Kontur stellt eine extreme Parametrisierung dar, bei der alle Abtastpunkte in einem Abschnitt liegen. Selbst wenn die Ausgangskontur exakt einer Ellipse entspricht, liefert das Ellipsenfitting eines Auschnittes eine andere Ellipse. Die Kontur muss deshalb vollständig sein und die Abtastpunkte sollten möglichst gleichmäßig verteilt sein.

16. Fourierdeskriptoren

16.1. Fourierdeskriptoren

Mit der Betrachtung von Konturen als komplexwertige Funktionen ist es möglich, die Fourierkoeffizienten dieser Funktion als Merkmale der Kontur zu betrachten. In dieser Anwendung werden die Fourierkoeffizienten als Fourierdeskriptoren bezeichnet. Beim Ellipsenfittung (Abschnitt 15.4) haben wir die Fourierkoeffizienten benutzt, um eine Ellipse zu fitten und die Formabweichung zu bemessen. Für allgemeine Merkmale zur Beschreibung der Form einer Kontur wünscht man sich in der Regel die Unabhängigkeit dieser Merkmale von der Lage, Orientierung und eventuell anderen Verfälschungen, die je nach Anwendung zum Beispiel durch die Bildaufnahme entstehen können. Mathematisch formuliert sollen die Merkmale invariant gegenüber bestimmten Transformationen der Kontur sein. Bei den Fourierdeskriptoren zeigt sich, dass diese Invarianz oft durch Normalisierung erreicht werden kann. Einige dieser Normalisierungschritte sollen nun näher betrachtet werden.

16.1.1. Translationen

Eine Translation in der komplexen Ebene läßt sich leicht durch

$$z_n^{[T]} = z_n + c \quad \text{mit} \quad n = 0, ..., N-1, c \in \mathbb{C}$$

beschreiben. Dabei ist c eine konstante komplexe Zahl, die die Translation beschreibt. Durch Anwendung der Fouriertransformation erhalten wir:

$$\alpha_k(z^{[T]}) = \alpha_k(z) + \alpha_k(c) = \alpha_k(z) + c\sqrt{N}\delta_k$$

Man sieht, dass alle Fourierkoeffizienten außer dem nullten Fourierkoeffizient unverändert bleiben und damit invariant gegenüber einer Translation sind. Der nullte Fourierkoeffizient beschreibt die Lage des Objektes, da er als Gleichanteil proportional zum Schwerpunkt des Objektes ist.

16.1.2. Isotrope Skalierungen

Eine isotrope Skalierung in der komplexen Ebene wird durch Multiplikation mit einer reellen Konstante beschrieben:

$$z_n^{[S]} = s \cdot z_n \quad \text{mit} \quad n = 0, ..., N-1, s > 0, s \in \mathbb{R}$$

Damit gilt für die Fourierkoeffizient $\alpha_k(z^{[S]}) = s \cdot \alpha_k(z)$. Im Prozess der Normalisierung suchen wir jetzt eine Skalierung s_{norm}, die einen Koeffizienten α_q auf einen festen Betrag setzt, zum Beispiel $\alpha_1(z^{[S]}) = 1$. s_{norm} berechnet sich dann zu

$$|\alpha_1(z^{[S]})| = s_{norm} \cdot |\alpha_1(z)| = 1 \rightarrow s_{norm} = \frac{1}{|\alpha_1|}$$

Setzen wir das berechnete s_{norm} nun in die Transformation ein, erhalten wir

$$\alpha_k(z^{[S]}) = s_{norm} \cdot \alpha_k(z) = \frac{\alpha_k}{|\alpha_1|}$$

als skalierungsinvariante Fourierdeskriptoren.

16.1.3. Rotation und Startpunktverschiebung

Als nächstes betrachten wir Rotationen mit dem Rotationswinkel β:

$$z_n^{[R]} = e^{i\beta} \cdot z_n \quad \rightarrow \quad \alpha_k(z^{[R]}) = e^{i\beta} \cdot \alpha_k(z) \quad \text{mit} \quad n = 0, ..., N-1$$

Da nur die Phase beeinflusst wird, gilt $|\alpha_k(z^{[R]})| = |\alpha_k(z)|$ - das Amplitudenspektrum ist rotationsinvariant. Zur Normalisierung benutzen wir die Phase eines Fourierkoeffizient α_q, die wir auf Null setzen. Wir könnten dazu auch den Fourierkoeffizient verwenden, den wir zur Normalierung der Skalierung eingesetzt haben: Bisher haben wir nur den Betrag normalisiert. Es bezeichne $\phi(z)$ die Phase von z, dann normalisieren wir

$$\phi(\alpha_q(z^{[R]})) = \phi(\alpha_q(z)) + \beta_{norm} + 2\pi l = 0 \quad \text{mit} \quad l \in \mathbb{Z}$$

Die Mehrdeutigkeit der Phase ist kein Problem, da bei der Rotation Vielfache von 2π keine Rolle spielen. Wir können also speziell $l = 0$ wählen, nach β_{norm} auflösen und einsetzen:

$$\alpha_k(z^{[R]}) = e^{-i\phi(\alpha_q)} \cdot \alpha_k(z) \quad \text{mit} \quad \forall\ k$$

Diese $\alpha_k(z^{[R]})$ sind rotationsinvariante Fourierdeskriptoren für die Kontur z.
Allerdings haben wir bisher vorausgesetzt, dass der Startpunkt der Konturen z_0 fest ist. Bei praktischen Verfahren zur Konturfolge und verschiedenen Lagen der Objekte ist dies nicht gegeben. Die Aufzählung der Konturpunkte beginnt an einer beliebigen Stelle. Wir müssen also dafür sorgen, dass die Fourierdeskriptoren invariant bezüglich einer solchen Startpunktverschiebung sind. Der Beginn mit einem verschobenen Startpunkt ist nichts anderes als eine Verschiebung der Funktion. Wir können formulieren:

$$z_n^{[Sh]} = (S_d z)_n = z_{n-d}$$

Mit dem Verschiebungstheorem und der Normierung der Phase eines Koeffizienten α_p zu Null, ergibt sich:

$$\alpha_k(z^{[Sh]} = \alpha_k(z) e^{2\pi i k \frac{d_{norm}}{N}} \ \rightarrow \ \phi(\alpha_p(z^{[Sh]}) = \phi(\alpha_p(z)) + 2\pi p \frac{d_{norm}}{N} + 2\pi l = 0$$

Wir lösen diese Gleichung nach der Verschiebung d_N auf

$$d_{norm} = -\frac{\varphi_p(z)N}{2\pi} + l\frac{N}{p}$$

und können alle Fourierdeskriptoren damit transformieren. Jetzt haben wir Startpunktinvariante Deskriptoren. Bei der Bestimmung der Verschiebung müssen wir wieder die Mehrdeutigkeit der Phasen beachten. Die ermittelte Verschiebung ist mehrdeutig mit einer Verschiebung $l\frac{N}{p}$. Die einfachste Möglichkeit, diese Mehrdeutigkeit zu umgehen, ist es, $p = 1$ zu verwenden. Die Verschiebung um N ist dann genau die Verschiebung um eine Periode und ändert somit nichts. Bei genauer Betrachtung fällt noch eine weitere Besonderheit auf: Die berechnete Verschiebung ist nicht ganzzahlig. Im Prinzip berechnen wir die Verschiebung für das trigonometrische Interpolationspolynom. Wenden wir diese Verschiebung auf die diskreten Fourierdeskriptoren an, erzeugen wir eine neue Abtastung der interpolierten Kurve. Wenn das Abtasttheorem eingehalten ist, ist das Ergebnis exakt. Praktisch ist es ausreichend, mit hinreichend vielen Punkten zu rechnen.
Für die Normalisierung der Startpunkt-Verschiebung dürfen wir nicht den Fourierkoeffizient verwenden, dessen Phase wir bei der Normalisierung der Rotation bereits auf Null gesetzt haben, es muss also $p \neq q$ sein. Da wir aber bei der Normalisierung der Startpunkt-Verschiebung auch die Phase des q-ten Koeffizienten transformieren, wird die Normierung bezüglich der Rotation zerstört. Rotations- und Startpunkt-Normalisierung scheinen sich zu widersprechen.
Dieses Problem können wir lösen, indem wir Rotation und Startpunktverschiebung gleichzeitig normalisieren. Wir verknüpfen beide Transformationen:

$$\alpha_k(z_n^{[R,Sh]}) = \alpha_k(z_n^{[R]})e^{2\pi ik\frac{d}{N}} = \alpha_k(z_n)e^{i\beta}e^{2\pi ik\frac{d}{N}}$$

Zur Normalisierung setzen wir die Phase zweier Fourierkoeffizienten auf 0:

$$\varphi_k(z) + \beta_{norm} + 2\pi k\frac{d_{norm}}{N} + 2\pi l = 0 \quad \text{für} \quad k = \{p, q\}$$

Die Lösung des Gleichungssystems liefert uns β_{norm} und d_{norm} und nach Einsetzen invariante Fourierdeskriptoren für Rotation und Startpunktverschiebung.

16.1.4. Parametrisierung

Mit den bisher betrachteten Normalisierungen können wir Fourierdeskriptoren bezüglich Transformationen T normalisieren, die sich aus Verschiebung, Rotation und Skalierung zusammensetzen. Grundannahme war aber, dass sich die Punkte nach der Transformation unmittelbar entsprechen, also $z_i^{[T]} = T(z_i)$ ist. Bei Berücksichtigung einer Startpunktverschiebung muss noch $z_i^{[T]} = T(z_{i-n})$ gelten. Das heißt, dass man bei der Abtastung jeweils exakt die gleichen (entsprechenden) Punkte treffen muss. Dies ist aber bei den meisten Verfahren nicht von vornherein gegeben. Bei der Konturfolge sind die Abtände der Punkte

unterschiedlich, je nach dem, ob der Konturabschnitt parallel zu den Achsen oder diagonal verläuft (Abbildung 16.1).

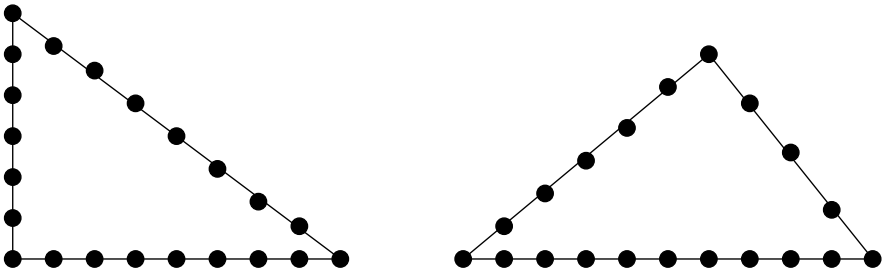


Abbildung 16.1.
Unterschiedliche Parametrisierung durch unterschiedliche Lage im Gitter

Um dies im Nachhinein zu korrigieren, kann man die Folge von Punkten in eine Kurve umwandeln und erneut digitalisieren. Eine mögliche Vorgehensweise wäre:

- Interpolation oder Approximation der Punktfolge durch eine Kurve
- Erneute Abtastung der Kurve mit konstantem Abstand (Bogenlänge)

Bei einer Skalierung würde dieses Vorgehen zu einer unterschiedlichen Zahl von Abtastpunkten führen. Dies ist für Ähnlichkeitstransformationen leicht lösbar: Man verteilt eine konstante Anzahl von Abtastpunkten über die Kurve, indem die Gesamtlänge der Kurve gleichmäßig aufgeteilt wird.

16.1.5. Alternative Amplitudenspektrum?

Da das viel einfacher zu berechnende Amplitudenspektrum invariant gegenüber Rotationen und Startpunktverschiebungen ist, wird dieses häufig als Merkmal zur Klassifikation benutzt. Dabei werden die Phasen alle ignoriert und es geht Information verloren. Wir wollen in einem Experiment klären, wie stark dieser Verlust ist. Wir wollen zu einer Kontur eine andere Kontur erzeugen, die das gleiche Amplitudenspektrum besitzt. Dazu erzeugen wir das Spektrum der Ausgangskontur, verändern die Phasen der Fourierkoeffizienten willkürlich und transformieren zurück. Die Abbildungen 16.2, 16.3 und 16.4 zeigen jeweils das Ausgangsobjekt und eine so erzeugte Kontur mit dem gleichen Amplitudenspektrum.

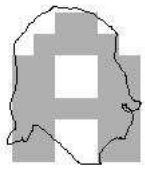


Abbildung 16.2.
Kontur mit gleichem Amplitudenspektrum wie das "A"

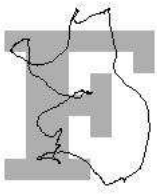


Abbildung 16.3.
Kontur mit gleichem Amplitudenspektrum wie das "F"

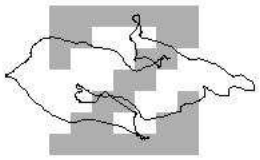


Abbildung 16.4.
Kontur mit gleichem Amplitudenspektrum wie das "Z'

Offensichtlich weisen sehr unterschiedliche Konturen das gleiche Amplitudenspektrum auf, die dann über diese Merkmale nicht zu trennen sind. Der Weg der Normalisierung dürfte also meist vorzuziehen sein.

16.1.6. Affine Transformationen

Eine affine Transformation in der komplexen Ebene lässt sich durch

$$z_n^{[aff]} = a \cdot z_n + b \cdot \overline{z_n} + c$$

beschreiben (13.13). Wir können die Translation c weglassen, da diese nur den nullten Fourierkoeffizient $\alpha_0(z)$ beeinflusst. Es folgt

$$\alpha_k^{[aff]} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} (a z_n + b\overline{z_n}) e^{-2\pi i k \frac{n}{N}} = a \cdot \alpha_k(z) + b \cdot \overline{\alpha_{-k}(z)}$$

Man kann sich nun für die Normalisierung zwei $\alpha_k(z)$ vorgeben und diese auf Null setzen und daraus a und b für die Transformation in die Standardlage berechnen. Das sieht recht einfach aus, schwieriger ist allerdings ein anderes Problem zu lösen: die affin invariante Parametrisierung der Kurve.

Teil VIII.

Signalbasierte 3D-Rekonstruktion

17. 3D-Rekonstruktion

In diesem Abschnitt soll auf keinen Fall eine Übersicht über 3D-Rekonstruktionsverfahren gegeben werden. In den meisten dieser Verfahren spielen signalbasierte Methoden nur die Rolle eines kleineren Werkzeuges in einem Teilschritt.
In einigen speziellen Verfahren sind aber signalbasierte Ansätze die Grundlage.
In der Computer-Tomographie (CT) liefern die Sensoren zunächst Bilder, die nicht direkt interpretierbar sind. Die gemessenen Intensitätswerte repräsentieren die Gesamtwirkung aller auf dem Strahlenweg liegenden Objekte. Erst eine entsprechende Aufbereitung erzeugt ein 3D-Bild. Dessen Schnitte lassen sich ansehen, das 3D-Bild lässt sich segmentieren und Objekte lokalisieren.
In Verfahren der Bildverarbeitung, die mit einem definierten Beleuchtungsmuster arbeiten (active vision), spielen auch signaltheoretische Überlegungen eine große Rolle. Besonders gilt dies bei Verwendung von regelmäßigen Gitter-Strukturen.

18. Computertomografie (CT)

Bei der Computertomografie werden Röntgenstrahlen benutzt, um das Körperinnere zu untersuchen. Die klassische Röntgenaufnahme arbeitet mit einer möglichst punktförmigen Röntgenstrahlenquelle. Der Körper wirft Schatten auf den Röntgenfilm oder -Sensor. Da die Strahlen den Körper teilweise durchdringen, zeichnen sich auch Objekte im Inneren des Körpers ab.

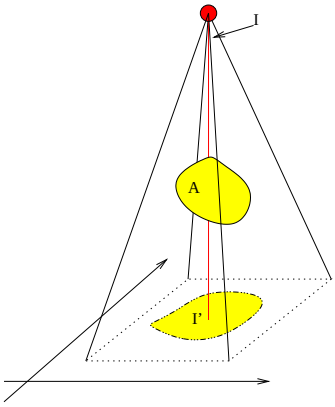


Abbildung 18.1.
Prinzip Röntgen

Um die Berechnungen einfach darzustellen, wollen wir vereinfacht annehmen, dass die Röntgenstrahlen parallel in Y-Richtung durchlaufen. Die Aufnahmefläche befinde sich in der x-z-Ebene. (Wir werden sehen, dass man bei geeigneten Aufnahmen diese parallelen Strahlen wirklich findet.)

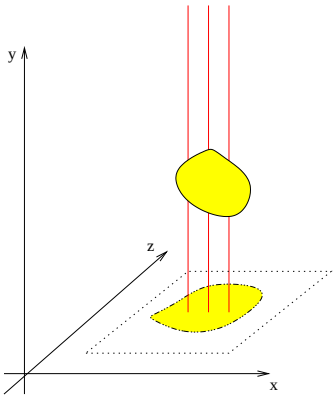


Abbildung 18.2.
Prinzip Röntgen

Die Absorption der Röntgenstrahlen und damit die auf der Aufnahmefläche auftreffende Intensität beschreibt das Lambert-Beer'sche Gesetz:

$$I(x,z) = I_0 e^{-\int \mu(x,y,z)dy}$$

Der Schwächungskoeffizient (Absorptionskoeffizient) μ wird durch die Eigenschaften des Materials bestimmt und ist abhängig vom Ort. Im folgenden soll stets der Logarithmus der Intensitäten betrachtet werden, da dies zu einer einfacheren Berechnung führt.

$$g(x,z) = \log I(x,z) = -\int \mu(x,y,z)\ dy \cdot \log I_0 = c \cdot \int \mu(x,y,z)\ dy = \int M(x,y,z)\ dy$$

Aus den 2-dimensionalen Schatten ist eine 3D-Rekonstruktion nicht möglich. Die Computertomografie verwendet Aufnahmen aus verschiedenen Winkeln.

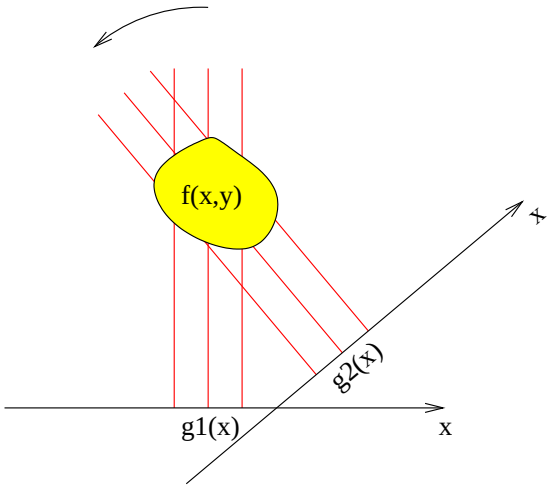


Abbildung 18.3.
Prinzip Computertomografie

18.1. Fourier-Slice-Theorem

Wir haben in der Darstellung 18.3 genau eine x-y-Ebene mit festem z betrachtet. Das wollen wir im folgenden auch in der formelmäßigen Darstellung verwenden. Für eine Aufnahme gilt

$$g(x) = \int_{-\infty}^{\infty} M(x,y)\ dy$$

Wir berechnen die Fourierkoeffizient der Funktion g:

$$\begin{aligned}\alpha_g(\nu_x) = \int_{-\infty}^{\infty} g(x)e^{\nu_x x}\,dx &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M(x,y)e^{\nu_x x}\,dy\,dx \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M(x,y)e^{\nu_x x+0y}\,dy\,dx \\ &= \alpha_M(\nu_x, \nu_y = 0)\end{aligned}$$

Die Fourierkoeffizient der Funktion g sind also die Fourierkoeffizient der Funktion $M(x, y)$ in der "Scheibe" (slice) $\nu_y = 0$. Diese Eigenschaft wird als Fourier-Slice-Theorem bezeichnet. Eine Aufnahme liefert somit eine Scheibe des Spektrums der Funktion $M(x, y)$. Könnten wir die restlichen Fourierkoeffizient ergänzen, würden wir durch Rücktransformation die gesuchte Funktion $M(x, y)$ erhalten. Dazu müssen wir uns ansehen, was die Aufnahmen mit der rotierten Anordnung liefern. Da eine Rotation der Funktion auch eine Rotation des Spektrums bewirkt, ermitteln die rotierten Aufnahmen rotierte Scheiben des Spektrums. Somit ist eine Zusammensetzung des Spektrums aus den Teilaufnahmen möglich. Praktisch sind natürlich noch Probleme wie eine Interpolation zu lösen.
Wir können den Algorithmus nun zusammenfassen:

- Wir erfassen die Projektionsfunktion senkrecht zu einer Projektionsgeraden. Senkrecht zur Projektionsgeraden werden dazu alle Absorptionswerte entlang der Projektionsstrahlen addiert und der Projektionsgeraden an diesem Punkte zugeordnet.
- Wir bilden das 1D-Spektrum dieser Projektionsfunktion. Dieses 1D-Spektrum ist gleichzeitig der Ausschnitt (slice) aus dem 2D-Spektrum längs einer Geraden, die die gleiche Orientierung hat wie die Projektionsgerade, aber durch den Koordinatenursprung geht.
- Durch Drehen des Sensors erzeugen wir nun "viele" slices des 2D-Spektrums.
- Durch Füllen der Lücken mittels Interpolation erhalten wir ein vollständiges 2D-Spektrum und durch die Rücktransformation erhalten wir $f(x, y, z)$ für ein z.
- Zur Berechnung der vollständigen 3D-Funktion $f(x, y, z)$ verfahren wir analog mit allen (x,y)-Ebenen.

18.2. Rückprojektion (Backprojection)

Es sei daran erinnert, dass wir hier logarithmische Werte für die Absorption betrachten und sich die gemessenen logarithmierten Werte als Summe/Integral der Absorptionswerte auf dem zugehörigen Strahl betrachten lassen. Berechnet man diese Summenwerte $g(r, \varphi)$ für ein Bild, wird dies auch als Radontransformation bezeichnet.

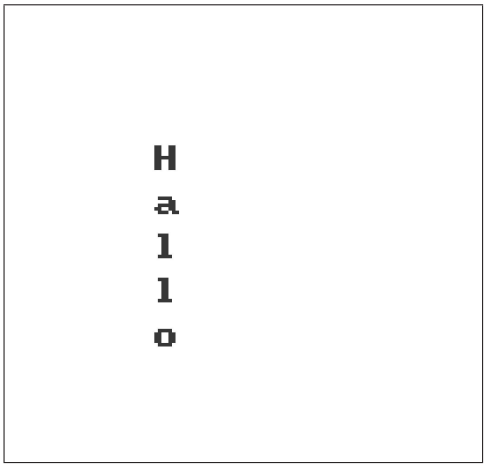
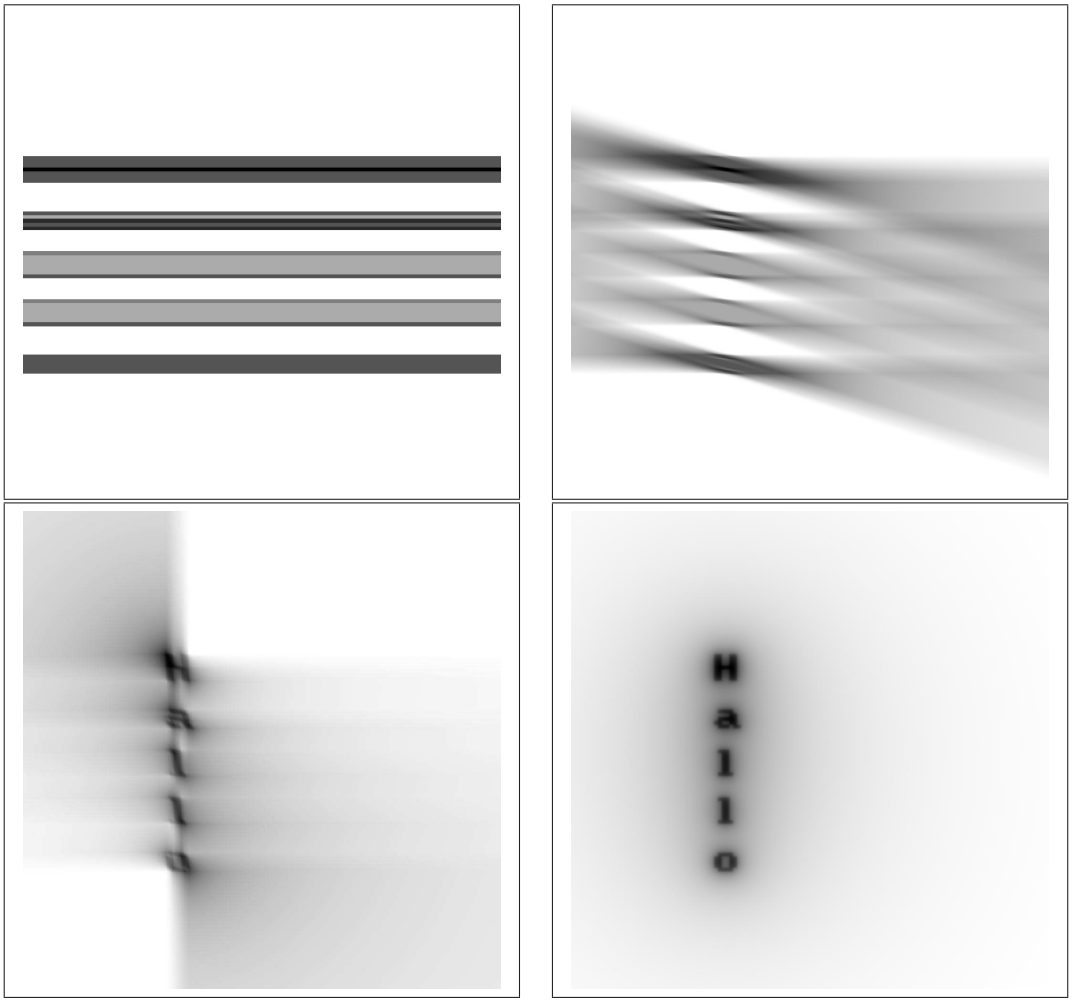


Abbildung 18.4.
Originalbild "Hallo"

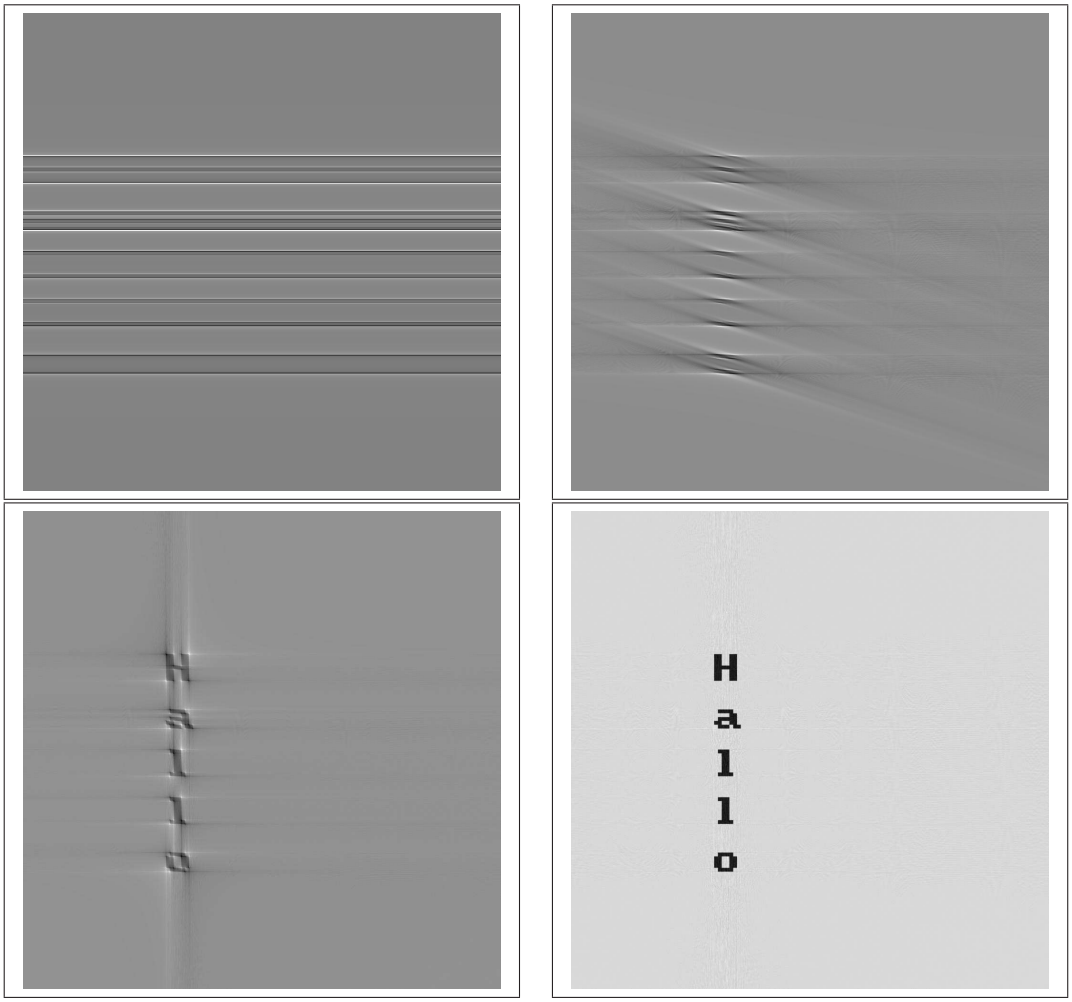
Abbildung 18.5.
Radon-Transformierte des Bildes "Hallo"

Eine naive Idee, die Radontransformation umzukehren, ist die Rückprojektion. Man zeichnet die zu den Funktionwerten $g(r, \varphi)$ gehörigen Geraden in das Bild ein, wobei die Grauwerte aller Geraden summiert werden.



Rückprojektion in 4 Teilschritten

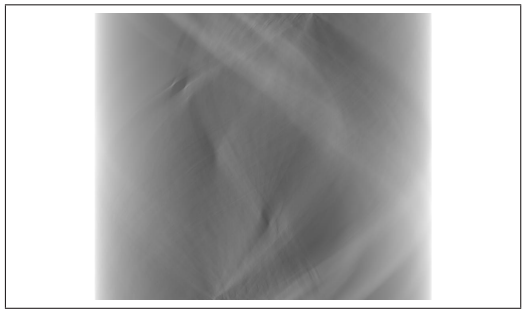
Offensichtlich entsteht dadurch nicht das Originalbild wieder. In dem speziellen Fall mit nur wenigen scharzen Punkten ist immerhin der Text wieder erkennbar. Man kann sich aber leicht überlegen was passiert: Jeder Punkt wird bei der Radontransformation durch ein Büschel von Strahlen repräsentiert, die sich bei der Rückprojektion schneiden und dort die höchste Intensität liefern. In der Umgebung fällt die Intensität umgekehrt proportional zum Abstand ab. Dieses Verhalten gilt für alle Punkte gleichermaßen: Bei Radontransformation und Rückprojektion handelt es sich also um einen LSI-Operator. Das legt nun aber nahe, dass es möglich ist, diesen zu invertieren und eine exakte Rekonstruktion zu erhalten. Diese gefilterte Rückprojektion ist eine Alternative zum Fourier-Slice-Theorem, um die 3d-Funktion aus einer Computer-Tomografie zu ermitteln. Es erweist sich, dass eine einfache Filterung der eindimensionalen Funktion $g(r, \varphi)$ für ein festes φ ausreicht.



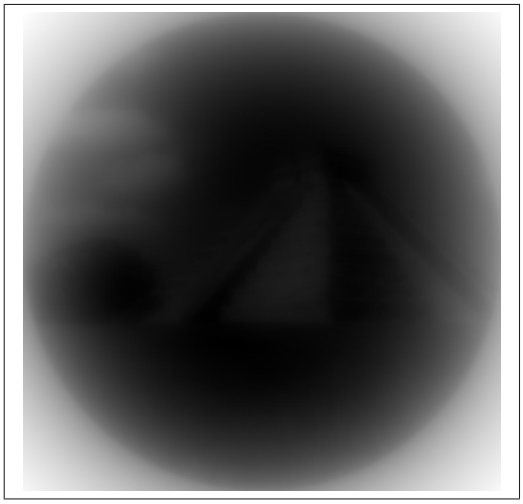
gefilterte Rückprojektion in 4 Teilschritten

Nehmen wir ein reales Bild. Wir beschneiden es hier kreisförmig mit zero-padding, so dass die Radon-Transformierte weniger durch die Ecken gestört wird.

Original



Radon-Transformierte



Rückprojektion

gefilterte Rückprojektion

19. Aktives Sehen

Bei der klassischen (Stereo-)Triangulation verwendet man die Bilder zweier Kameras, die seitlich gegeneinander versetzt sind und die gleiche Szene betrachten. Hat man die Kameras kalibriert, ist es möglich einem Bildpunkt einen Sehstrahl zuzuordnen. Die Triangulation sucht nun für einzelne Punkte des zu vermessenden Objekts in beiden Bildern die Projektionen (korrespondierende Punkte).
Aus den beiden Sehstrahlen lässt sich nun ein Raumpunkt ermitteln. Schneiden sich die berechneten Sehstrahlen durch Messfehler nicht, wird ein Punkt ermittelt, der zu beiden Strahlen den geringsten Abstand hat. Dies wird auch "Vorwärts-Projektion" genannt. Alternativ kann man auch den Raumpunkt suchen, dessen Projektionen in die Bilder den kleinsten Abstand zu den korrespondierenden Punkte aufweisen ("Rückwärts-Projektion"). Diese Verfahren lassen sich auch mit mehr als zwei Kameras durchführen.
Das Hauptproblem ist hier in der Praxis die Ermittlung der Projektionen eines Raumpunktes, das Korrespondenz-Problem. Das "Aktive Sehen" löst dieses Problem, indem eine Kamera durch eine steuerbare Lichtquelle ersetzt wird. Identifiziert man im Bild die Wirkung eines Strahls der Lichtquelle, kennt man sowohl die Eigenschaften des Lichtstrahls als auch den Sehstrahl der Kamera und kann damit triangulieren.

19.1. Projektion von Lichtstrahlen

Die einfachste Art der Triangulation geschieht dadurch, dass ein Gerät einen Lichtstrahl in die Szene projiziert. Der Ausgangspunkt des Lichtstrahls und sein Richtungsvektor sind bekannt. Das Auftreffen des Lichtstrahls auf die Oberfläche des Objektes wird im Bild der Kamera als Lichtfleck identifiziert. Durch Triangulation kann damit der 3D-Punkt im Weltkoordinatensystem berechnet werden. Als Lichtquelle wird häufig eine Laserdiode mit Kollimatoroptik verwendet. Um alle Objektpunkte vermessen zu können, muss der Lichtstrahl zweidimensional ablenkbar sein. Pro Punkt ist eine Aufnahme nötig. Das ist ein enormer Aufwand.

19.2. Projektion mit Lichtebenen

Der Aufwand lässt sich reduzieren, wenn statt eines Lichtstrahles eine Lichtebene projiziert wird, die sich über die Szene schwenken lässt. Im Bild "sieht" man dann eine Linie der Lichtebene. Kennt man Position und Eigenschaften des Projektors, ist damit die Gleichung der Lichtebene bekannt. Die 3D-Punkte lassen sich berechnen, indem für jeden Punkt der

beleuchteten Linie im Bild der Sehstrahl und dessen Schnitt mit der Lichtebene berechnet wird. So können wir mit einer Aufnahme alle 3D-Punkte rekonstruieren, die mit dieser Lichtebene beleuchtet wurden. Dies tun wir nun für alle weiteren Lichtebenen des Projektors. Wir brauchen also genauso viele Bilder wie Lichtebenen - für eine vernünftige Auflösung immer noch ein sehr hoher Aufwand.

19.3. Kodierter Lichtansatz

Um die 3D-Rekonstruktion weiter zu beschleunigen, versuchen wir nun, viele Lichtebenen gleichzeitig in das Bild zu projizieren. Die Zuordnung der hellen Bildpunkte zur richtigen Lichtebene gelingt jetzt aber nur, wenn wir wenige Ebenen projizieren. Dann ist auch die Auflösung gering.
Immer noch schneller als die Projektion vieler Ebenen einzeln ist es, wenn wir nacheinander verschiedene Muster projizieren, die uns die Unterscheidung der Ebenen ermöglichen. (Für bewegte Objekte ist es damit allerdings noch nicht geeignet.) Wir brauchen also einen Projektor, dessen "Lichtebenen" sich steuern lassen. Dies kann im Prinzip wie in Abb. 19.1 aussehen.

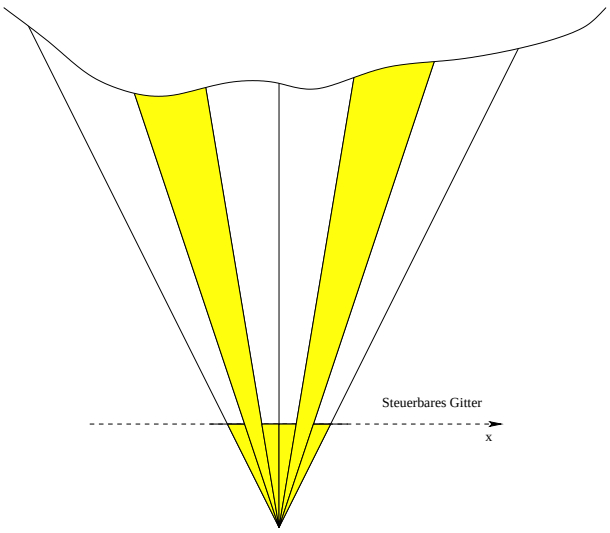


Abbildung 19.1.: Projektionsanordnung

Das steuerbare Gitter besteht aus Streifen, die einzeln hell oder dunkel geschaltet werden können. Die durch die Streifen erzeugten "Lichtebenen" haben eine gewisse Dicke, wir wollen deshalb auch die projizierten Lichtflächen als Streifen bezeichnen. Ein Videoprojektor ("Beamer") ist ein solches Gerät, wenn wir Pixel einer Spalte oder Zeile zu einem Streifen zusammenfassen.
Wir wollen die Streifen derart hell und dunkel steuern, dass wir in einer Sequenz von Bildern jedem Pixel einen Streifen zuordnen können. Wenn wir in der Sequenz einen einzelnen

Punkt betrachten, werden wir eine sich ändernde Helligkeit wahrnehmen, wobei der zeitliche Verlauf der Steuerung der Streifen im Projektor folgt. Wir können nun daraus auf den Streifen des Gitters zu schließen, indem sich der Punkt befindet.
Dazu muss jeder Streifen einen eindeutigen Helligkeitsverlauf aufweisen. Wir können die binären Zustände "dunkel" oder "hell" zur Kodierung der Nummer des Streifens nutzen. Für n Streifen benötigen wir folglich $\log_2 n$ Bit. Mit $\log_2 n$ Bildern können wir für jedes einzelne Pixel entscheiden, welcher von den n verschiedenen Streifen des Projektors das Pixel "getroffen" hat. Zur Vereinfachung demonstrieren wir dies am Beispiel eines Projektors mit nur 8 Streifen.

Streifen	0	1	2	3	4	5	6	7
t=0	0	0	0	0	1	1	1	1
t=1	0	0	1	1	0	0	1	1
t=2	0	1	0	1	0	1	0	1

Die Zahlen in der jeweiligen Spalte sind die Digits der Streifennummer im Dualkode. Wir beleuchten also das Objekt dreimal, wobei jeder Streifen entsprechend der Zeile unserer Matrix hell (1) oder dunkel(0) ist. Wir brauchen also nur drei Bilder und können eindeutig jedem Pixel einen der acht Kodes zuordnen. Bei 256 Streifen des Projektors müssen wir also nur 8 Muster projizieren und damit nicht mehr jeden Streifen einzeln.
Entscheidend ist nun die Erkennung, ob ein Pixel im jeweiligen Muster beleuchtet ist oder nicht. Da sich die Oberflächeneigenschaften des Objektes, aber auch die Beleuchtungsverhältnisse ändern, müssen sinnvolle lokale Binarisierungsstrategien eingesetzt werden. Pixel können aber auch im "Schatten" liegen, also in einem Bereich, der von der Lichtquelle aus nicht gesehen wird. Diese Pixel müssen ebenfalls erkannt und aus der Rekonstruktion ausgeschlossen werden. Dies legt nahe, den Kode 0 (alles dunkel) nicht zu verwenden, da sich dieser vom Schatten nicht unterscheidet.
Eine einfache Methode, um die Hell-/Dunkel-Erkennung sicher zu machen, ist die Verdopplung der Anzahl der Muster: Jedes Muster wird einmal direkt (z.B. 0110) und einmal invertiert verwendet (also 1001). Dann muss sich die Helligkeit jedes beleuchteten Pixels umkehren. Bildet man die Differenz dieser Bilder, so wird es Pixel mit positiver und negativer Differenz geben, die damit eindeutig als "0" oder "1" identifizierbar sind, und Pixel mit Differenzen nahe Null, die nicht vom Projektor beleuchtet werden und somit bei der 3D-Rekonstruktion ausgelassen werden müssen. Der Preis für die hohe Sicherheit ist natürlich eine Verdopplung des Zeitaufwandes.
Eine unsichere Hell-/Dunkel-Entscheidung wird vorrangig im Bereich des Überganges zwischen zwei Streifen auftreten. Dabei kann es dazu kommen, dass ein Bitwert von dem linken, die anderen Bitwerte von dem rechten Streifen stammen. Die im Dualkode daraus resultierenden Fehler sind gravierend:

Streifen 10	$\underline{1}$	$\underline{0}$	1	$\underline{0}$
Streifen 11	1	1	$\underline{0}$	0
erkannt: 8	1	0	0	0

Eine Kodierung, bei der sich zwischen aufeinanderfolgenden Zahlen nur ein Bit ändert, vermeidet dies: Nur dieses eine Bit könnte falsch detektiert werden und der resultierende Wert ist dann der des Nachbarn. Da dies sowieso nur an der Grenze zwischen diesen Streifen auftreten wird, ist der Fehler geringfügig. Ein Kode mit dieser Eigenschaft ist der Gray-Kode. Ein 3-Bit Gray-Kode sieht so aus

Ebene	0	1	2	3	4	5	6	7
Bit 0	0	0	0	0	1	1	1	1
Bit 1	0	0	1	1	1	1	0	0
Bit 2	0	1	1	0	0	1	1	0

Ein Kode-Wort der Länge n des Binärkodes bestehe aus den Bits $b_1 b_2 ... b_n$ und das zugehörige Kodewort des Gray-Kodes aus den Bits $g_1 g_2 ... g_n$. Dann gilt die Bijektion

$$b_i = g_1 + g_2 + ... + g_i \quad (mod\ 2)$$

$$g_i = b_{i-1} + b_i \quad (mod\ 2)$$

Wie können wir die Zuordnung der Pixel zu Streifen zur Rekonstruktion nutzen? Benutzen wir einen Streifen näherungsweise als Ebene, so haben wir zu jedem Pixel den Sehstrahl und die Gleichung dieser Ebene. Da unsere "Lichtebenen" aber recht dick sind, ist der Fehler oder Unsicherheit einer solchen Rekonstruktion groß. Wir haben aber auch noch eine bessere Möglichkeit: Wir betrachten die Übergänge zwischen zwei Streifen als Ebene und verwenden die Bildpunkte, die im Bild am Übergang zwischen zwei Streifen liegen. Günstigenfalls kann man den Übergang zwischen den Streifen durch Interpolation in den aufgenommenen Mustern auch Subpixel-genau berechnen.
Die Dichte der Punktwolke der 3D-Rekonstruktion hängt damit von der Anzahl der Lichtebenen des Projektors ab.

19.4. Phasenshift-Verfahren

Wir ersetzen unser steuerbares Gitter in Abbildung 19.1 durch ein ideal periodisches Gitter. Das durchgelassene Licht ist dann eine periodische Funktion von x: $g(x) = g(x + \Delta x)$. Wir wollen wieder ein Reihe Bilder aufnehmen und dabei das Gitter verschieben. Wir betrachten n gleichgroße Schritte, bei denen das Gitter genau um Δx verschoben wird. Wenn wir in den aufgenommenen Bildern ein Pixel betrachten, dann sehen wir jeweils das Licht, welches an einem Punkt x durch das Gitter gelangt. Das entspricht bis auf einen Vorfaktor c für die Reflexionseigenschaften des Objektpunktes der Abtastung der Gitterfunktion g in n äquidistanten Schritten:

$$I_k = c \cdot g(x + k \cdot \frac{\Delta x}{n})$$

Unter realen Messbedingungen werden wir oft noch eine weitere Störung berücksichtigen müssen: Raumlicht, welches zusätzlich zum Projektor das Objekt beleuchtet und für eine Grundhelligkeit I_b sorgt.

$$I_k = c \cdot g(x + k \cdot \frac{\Delta x}{n}) + I_b$$

Ziel der Rekonstruktionsaufgabe wäre es nun, möglichst genau die Verschiebung x aus diesen Abtastpunkten zu rekonstruieren. Wenn wir hierfür wie beim kodierten Lichtansatz ein Rechteckgitter ansetzen, ist die Genauigkeit beschränkt auf das Intervall zwischen zwei Abtastpunkten, da eine genauere Lokalisation des Überganges von Hell zu Dunkel nicht möglich ist.
Bei einer anderen analogen Funktion, die nicht nur zwei verschiedene Funktionswerte aufweist, ist sicher mehr möglich. Bei beliebigen Gitter-Funktionen ist dies aufwendig und die Eindeutigkeit der Zuordung der analogen Gitterfunktion zu den diskreten Abtastpunkten ist nicht sicher.
Betrachten wir aber mal diese Problemstellung genauer, so erkennen wir ihre Verwandschaft mit dem Abtasttheorem: Auch wenn die Rekonstruktion der Funktion aus den Abtastpunkten nicht notwendig ist, würde es jedoch nicht schaden, es zu können - wir könnten dann die gesuchte Verschiebung x recht einfach ermitteln. Nun sagt uns aber das Abtasttheorem genau, wann wir das können: Wenn die Funktion bandbegrenzt ist und nur n Fourierkoeffizienten ungleich 0 besitzt, so reichen eben n Abtastpunkte aus. Wir wählen also n möglichst klein. Der nullte Fourierkoeffizient wird nicht ausreichen: Bei einer konstanten Funktion lässt sich keine Verschiebung bestimmen. Zudem wird der nullte Fourierkoeffizient durch die Raumhelligkeit verfälscht. Wir müssen also mindestens den ersten (und minus ersten) Fourierkoeffizient dazunehmen und untersuchen die Funktion $g(x) = a_0 + a_1 \cos(2\pi \frac{x}{\Delta x})$:

$$\begin{aligned} I_k &= c \cdot (a_0 + a_1 \cdot \cos(2\pi \frac{x + k \cdot \frac{\Delta x}{n}}{\Delta x})) + I_b \\ &= c \cdot a_0 + I_b + c \cdot a_1 \cdot \cos(2\pi \frac{x}{\Delta x} + k \cdot \frac{2\pi}{n}) \\ &= a_0^* + a_1^* \cdot \cos(\varphi + k \cdot \frac{2\pi}{n}) \end{aligned} \tag{19.1}$$

Die Gleichungen haben drei Unbekannte: a_0^*, a_1^* und φ, wobei uns eigentlich nur $\varphi = 2\pi \frac{x}{\Delta x}$ interessiert. a_1^* können wir als Maß dafür verwenden, wieweit Licht des Projektors auf diesen Punkt wirkt, ob φ also überhaupt sinnvoll zu bestimmen ist.
Da wir mindestens 3 Gleichungen benötigen, wählen wir $n = 3$ und 19.1 wird zu

$$\begin{aligned} I_0 &= a_0^* + a_1^* \cdot \cos(\varphi) \\ I_1 &= a_0^* + a_1^* \cdot \cos(\varphi + \frac{2\pi}{3}) \\ I_2 &= a_0^* + a_1^* \cdot \cos(\varphi - \frac{2\pi}{3}) \end{aligned} \tag{19.2}$$

Wir ermitteln die gesuchten Unbekannten zu:

$$\begin{aligned} a_0^* &= \tfrac{1}{3}(I_0 + I_1 + I_2) \\ a_1^* &= \tfrac{1}{3}\sqrt{(2 \cdot I_0 - I_1 - I_2)^2 + 3(I_1 - I_2)^2} \\ \varphi &= \arctan(\sqrt{3} \tfrac{I_1 - I_2}{2I_0 - I_1 - I_2}) \end{aligned} \tag{19.3}$$

Natürlich bleibt die Frage, wie ein Projektor aussehen soll, der eine Cosinus-Funktion und ihre verschobenen Funktionen projizieren kann. Der Video-Projektor mit seinem steuerbaren Gitter genügt nun nicht mehr, da die von ihm projizierten Muster scharfe Kanten und keine Übergänge enthalten. Nun gibt es aber verschiedene Möglichkeiten, die Abbildung des Projektors unscharf zu machen, einfach durch Defokussieren der Optik oder einen optischen Tiefpass im Strahlengang. Die sonst meist ungewollte Unschärfe hätte hier den gewünschten Effekt eines Tiefpasses, der von einem (symmetrischen) Rechteckmuster nur die Grundschwingung übrig lässt.
Wählen wir ein Grundraster von 4 Streifen im Projektor und schalten 2 Streifen hell und 2 Streifen dunkel, so können wir 4 Verschiebungen erzeugen, die nach dem Tiefpass eine gute Näherung einer verschobenen Cosinus-Funktion darstellen (siehe Grafik 19.2).

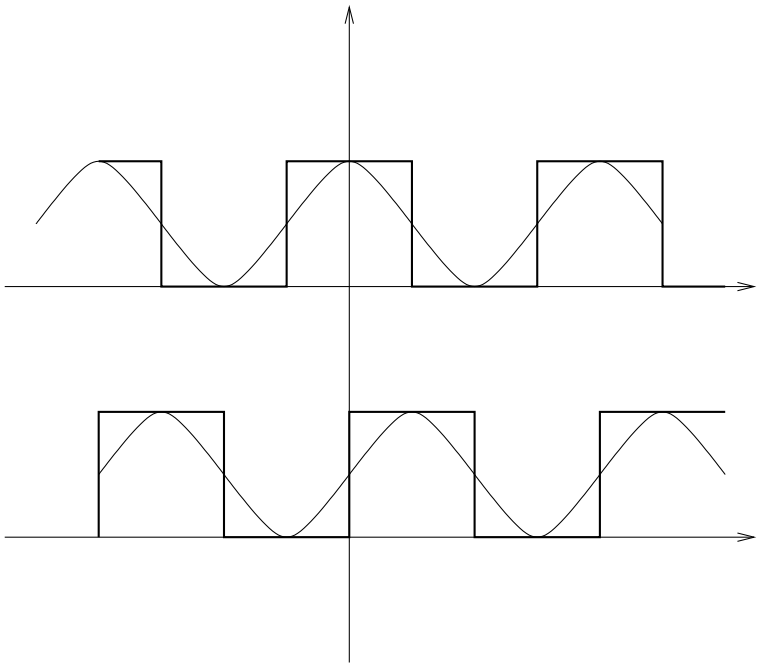


Abbildung 19.2.: Zwei Rechteckfunktionen und die entstehenden sinusförmigen Funktionen nach dem Tiefpass

Verwenden wir 4 Aufnahmen ($n = 4$) vereinfachen sich auch die Bestimmungsgleichungen:

$$\begin{aligned}
I_1 &= a_0^* + a_1^* \cdot \cos(\varphi) \\
I_2 &= a_0^* + a_1^* \cdot \cos(\varphi + \frac{\pi}{2}) = a_0^* - a_1^* \cdot \sin(\varphi) \\
I_3 &= a_0^* + a_1^* \cdot \cos(\varphi + \pi) = a_0^* - a_1^* \cdot \cos(\varphi) \\
I_4 &= a_0^* + a_1^* \cdot \cos(\varphi + \frac{3}{2}\pi) = a_0^* + a_1^* \cdot \sin(\varphi) \\
\sin(\varphi) \cdot a_1^* &= \frac{1}{2}(I_4 - I_2) \\
\cos(\varphi) \cdot a_1^* &= \frac{1}{2}(I_1 - I_3) \\
\varphi &= \operatorname{atan2}(I_1 - I_3, I_4 - I_2) + k \cdot 2\pi
\end{aligned}$$

Da wir nach dem Abtasttheorem die Originalfunktion genau rekonstruieren könnten, können wir davon ausgehen, dass die ermittelte Phase (theoretisch) exakt ist. Die Verschiebung x können wir somit subpixelgenau ermitteln und daraufhin den 3D-Punkt sehr genau rekonstruieren. Praktisch schränken natürlich Messfehler die Genauigkeit ein, eine Quantisierung der Phase durch das Pixelraster gibt es jedoch nicht.
Da der Phasenwert nur innerhalb einer Periode bestimmbar ist, kann der absolute Phasenwert so nicht bestimmt werden. Deshalb benutzt man eine Kombination des kodierten Lichtansatzes zur Bestimmung der Periode und des Phasenshiftverfahrens zur Ermittlung der Phase innerhalb der Periode.

19.5. Phasenbestimmung aus einer Aufnahme

Die bisher gezeigten Verfahren basieren auf der Projektion mehrerer Muster. Dabei darf sich das zu untersuchende Objekt zwischen zwei Aufnahmen nicht bewegen. Bei einem sich bewegenden Objekt könnte es immerhin möglich sein, eine Aufnahme mit einem Beleuchtungs-Muster vorzunehmen. Da das Beleuchtungs-Muster in der Aufnahme wiedererkennbar sein muss, kann es nicht kompliziert sein: Wir wählen ein sinusförmiges senkrechtes Gitter. Damit wir dieses im Bild rekonstruieren können, dürfen im sonstigen Bild entsprechende Muster nicht vorkommen. Ein Streifenmuster auf der Objektoberfläche ist vom Beleuchtungsmuster nicht zu unterscheiden. Einfach gesagt: die Oberfläche sollte möglichst farblich homogen sein. Weiterhin muss die Oberfläche möglichst glatt sein, Kanten durchbrechen das sichtbare Beleuchtungsmuster und behindern die Detektion.
Wenn diese Bedingungen erfüllt sind, sieht man im Bild das Muster der Beleuchtung und kann versuchen, wie bei den anderen Verfahren die Verschiebung zu bestimmen.
Als ersten Ansatz betrachten wir zu jedem Bildpunkt eine Umgebung, die der Größe der Periode des sichtbare Gitters entspricht. Damit liegt eine Periode in dieser Umgebung. Im Spektrum entspricht also der erste Fourierkoeffizient unserem Gitter. Die Verschiebung ist damit aus der Phase φ ermittelbar. Dazu muss gar nicht das vollständige Spektrum bestimmt werden.

In einem zweiten Ansatz betrachten wir das Spektrum des ganzen Bildes. Die gesuchte Information aus dem projizierten Gitter konzentriert sich in wenigen Fourier-Koeffizienten um die Gitterfrequenz. Wäre die Oberfläche total eben und parallel zur Bildebene, hätten wir ein regelmäßiges Gitter mit genau einer Frequenz. Bei sich ändernder Tiefe verschieben sich die Gitterstrukturen und es treten um die Gitterfrequenz weitere Komponenten auf. Genau genommen enthalten gerade diese Abweichungen die gesuchten Informationen über die Oberflächenstruktur des Objektes. Nachdem wir also entsprechend gefiltert haben, müssen wir versuchen, die Phase, also die Verschiebung zu rekonstruieren. Das könnten wir wie im ersten Ansatz durch Integration über eine passende Umgebung erreichen. Wir erinnern uns aber daran, wie sich die reellwertige Cosinus-Funktion aus zwei e-Funktionen zusammensetzt:

$$cos\varphi = \frac{1}{2}(e^{i\varphi} + e^{i-\varphi})$$

Die komplexe Fouriertransformation liefert uns diese Zerlegung in Form der zueinander konjugiert komplexen Fourierkoeffizienten α_k und α_{-k} und damit einen entscheidenden Vorteil: ein komplexer Wert lässt sich eindeutig einem Wert φ in der e-Funktion zuordnen, was bei der Cosinus-Funktion schwierig ist. Dies können wir nun folgendermaßen nutzen:

- Wir bestimmen das Spektrum unseres aufgenommenen Bildes
- Wir bestimmen (wenn nicht schon bekannt) die Frequenz unseres Gitters im Bild
- Wir filtern das Spektrum, so dass nur die Koeffizienten um die Gitterfrequenz übrig bleiben
- Wir setzen eine Halbebene auf Null, so dass nur eine der zusammengehörigen e-Funktionen übrig bleibt.
- Wir transformieren das Bild zurück und bestimmen für jeden Punkt aus Real- und Imaginärteil die Phase φ.

Wir haben damit zu unserem gefilterten Bild einen Imaginär-Teil erzeugt, der uns dann bei der Phasen-Bestimmung hilft. Dieses Vorgehen entspricht der Hilberttransformation des Real-Teils, was uns den Imaginär-Teil liefert.

19.6. Moire-Technik

Die Moire-Technik ist eine Rekonstruktionsmethode, die eine Gewinnung von Tiefeninformation mit Hilfe der klassischen Fotografie erlaubt. Dazu durchläuft das Licht ein Gitter auf dem Weg von der Lichtquelle zum Objekt und vom Objekt zur Kamera (Prinzipskizze in Abbildung 19.3).

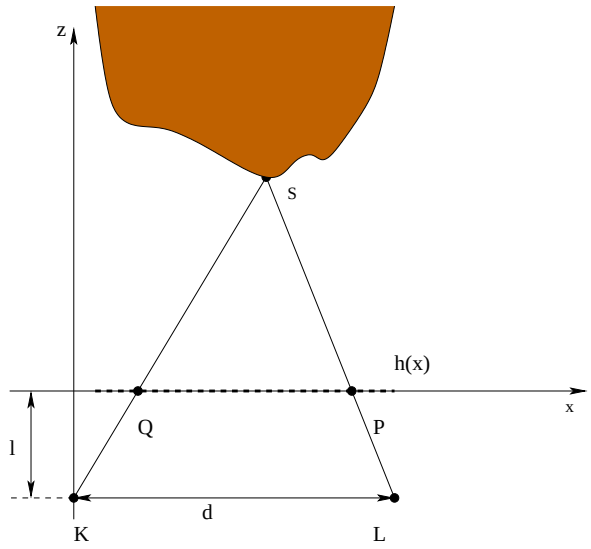


Abbildung 19.3.: Moire - Prinzipskizze

Eine Objektoberfläche wird durch die punktförmige Lichtquelle L beleuchtet und durch eine Kamera K auf eine Bildebene abgebildet. Dabei ist das Koordinatensystem so orientiert, dass die optische Achse der Kamera mit der z-Achse übereinstimmt. Die Kamera und die Lichtquelle die gleiche (negative) z-Koordinate $z_K = z_L = -l$ besitzt. Es ist $K = (0, 0, -l)$ der Ort des Projektionszentrums der Kamera und $L = (d, 0, -l)$ der Ort der Lichtquelle. Ein Oberflächenpunkt $S = (x, y, z)$ wird in den Punkt $B = (u, v)$ der Bildebene abgebildet, wobei gilt

$$u = \frac{f}{z+l} \cdot x \quad \text{und} \quad v = \frac{f}{z+l} \cdot y$$

Hier ist f die bekannte Bildweite der Kamera. Ist dann die Tiefe z eines 3D-Punktes bekannt, dann können aus den Bildkoordinaten u, v auch die x, y-Koordinaten des 3D-Punktes berechnet werden. Um Information über die Tiefe zu erhalten, legen wir in die x-y-Ebene ein periodisches Liniengitter, dessen Linien parallel zur y-Achse verlaufen sollen. Bei der Betrachtung eines Objektpunktes S wird nun:

- Die Beleuchtung des Punktes S durch das Gitter beeinflusst, indem das Licht der Lichtquelle das Gitter im Punkt P durchläuft und in Abhängigkeit von der Gitterfunktion geschwächt wird.

- Der von Punkt s reflektierte Lichtstrahl durchläuft auf dem Weg zur Kamera das Gitter im Punkt Q, was zu einer Schwächung in Abhängigkeit von der Gitterfunktion führt.

Für $Q = (x_Q, y_Q, 0)$ gilt

$$\frac{x_Q}{l} = \frac{x}{z+l} \quad \text{und} \quad \frac{y_Q}{l} = \frac{y}{z+l}$$

und damit

$$x_Q = \frac{l}{z+l} \cdot x \quad \text{und} \quad y_Q = \frac{l}{z+l} \cdot y$$

Hat P die Koordinaten $(x_P, y_P, 0)$, so gilt

$$x_P = x_Q + \frac{z}{z+l} \cdot d \quad \text{und} \quad y_P = y_Q = \frac{l}{z+l} \cdot y$$

Es sei nun h(x) die Durchlässigkeit des Gitters an der Stelle x. Dann wird die Helligkeit $g(u, v)$ des Bildpunktes im Bild durch das Produkt

$$h(x_Q) \cdot h(x_P) = h(x_Q) \cdot h(x_Q + \frac{dz}{z+l}) = h(\frac{lx}{z+l}) \cdot h(\frac{lx+dz}{z+l})$$

bestimmt (Die Stärke der Lichtquelle und die Reflexionseigenschaften der Oberfläche betrachten wir als konstanten Faktor, den wir hier vernachlässigen). Die Grautöne im Bild $g(u, v)$ bei hängen bei fester Abbildungsgeometrie von der Tiefe z ab, aber auch von der Lage des Punktes im Bild (x). Um diese Abhängigkeit zu eliminieren, können wir das Gitter verschieben, so dass in jedem Bildpunkt alle Gitterpunkte irgendwann einmal zum Tragen kommen. Die fotografische Aufnahme mittelt über diesen Vorgang

$$\begin{aligned} g^m(x, y, z) &= \int_X h(x_Q + v) \cdot h(x_Q + \frac{zd}{z+l} + v) dv \\ &= (h \circ h)(\frac{zd}{z+l}) \end{aligned}$$

Die Helligkeit eines Punktes hängt nur noch von z ab, die Funktion ist die Autokorrelation von h. In der Aufnahme detektierbar sind vor allem die Maxima als helle Streifen im Bild. Wir wissen, dass die Autokorrelation ihr Maximum im Ursprung hat. Die zu den Maxima gehörigen Tiefen sind also

$$\frac{d \cdot z_n}{z_n - l} = n \cdot X$$

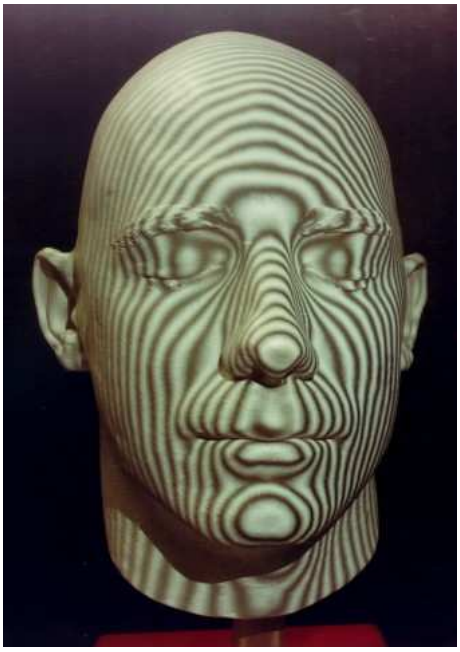


Abbildung 19.4.: Moire-Streifen sind Isolinien

Alle Punkte eines Maximums besitzen die gleiche Tiefe, siehe Abbildung 19.4. Die nächste Aufgabe ist die Zuordnung der einzelnen Streifen zu den zugehörigen Maxima z_n. Erforderlich ist dazu a-priori-Wissen. Das wird vor allem bei glatten Flächen ohne abrupte Übergänge und Kanten funktionieren.
Es gibt eine Reihe von modifizierten Moire-Verfahren in der Literatur. Statt eines Gitters kann man für Kamera und Lichtquelle zwei getrennte Gitter mit der gleichen Periode nutzen. Dann geht unsere Autokorrelationsfunktion einfach in die Kreuzkorrelation über und alle wesentlichen Eigenschaften bleiben erhalten.
Die Moire-Technik spielte in "analogen Zeiten" eine größere Rolle als heute: Alles dazu Notwendige funktioniert auch ohne Computer und mit analoger fotografischer Aufnahmetechnik.

20. Shape from focus/defocus

20.1. Shape from focus

Die Idee aus der Schärfe von Bildern die Tiefe zu berechnen, ist sehr einfach, siehe auch [33]. Dazu benutzen wir das Modell einer dünnen Linse, siehe Abb. 20.1. Es sei f die

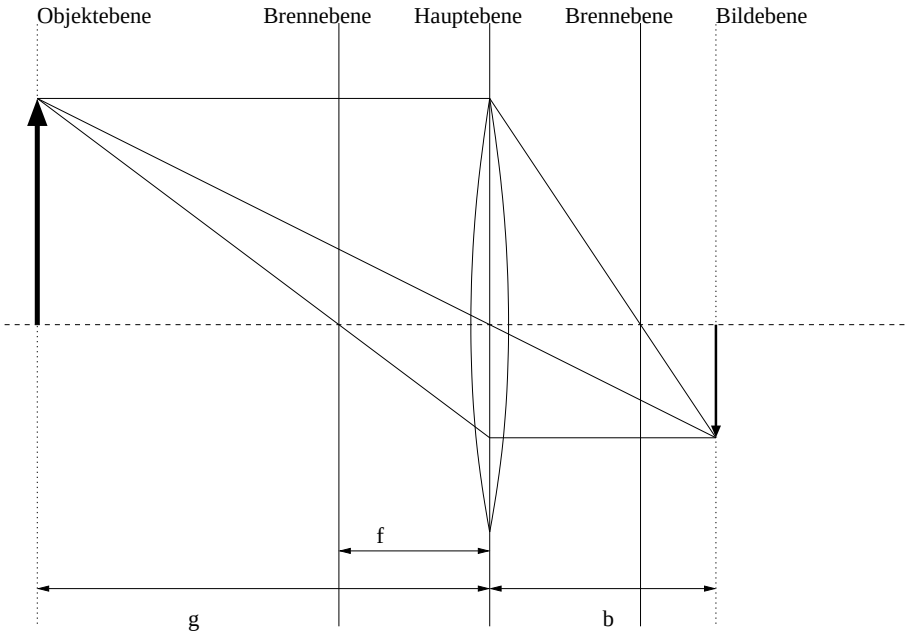


Abbildung 20.1.: Modell - dünne Linse

Brennweite der Linse, b die Bildweite, der Abstand des Projektionszentrum oder der Linse zur Bildebene, und g die Gegenstandsweite, der Abstand der Objektebene zur Linse, dann gilt

$$\frac{1}{f} = \frac{1}{b} + \frac{1}{g} . \tag{20.1}$$

Wird wie in der Abb. 20.1 ein 3D-Punkt scharf abgebildet, dann kann man mit bekannten f und b die Tiefe g ausrechnen. Wir müssen folglich für jedes Pixel feststellen, ob es scharf abgebildet ist oder nicht. Dazu benötigen wir Schärfe-Maße, die im Abschnitt **??** untersucht werden. Es ergeben sich nun eine Reihe praktischer Probleme:

- Da immer nur eine Ebene ermittelt werden kann, muss systematisch entweder das Objekt verschoben oder die Kameraeinstellung geändert werden. Dies muss quasi kontinuierlich erfolgen, um eine vernünftige Auflösung des Tiefenwertes zu erhalten. Das

ist praktisch schwer möglich, so dass die Veränderung in wenigen diskreten Schritten erfolgt. Es sind so zum Beispiel 15 Aufnahmen zu erstellen, um 15 unterschiedliche Tiefenwerte unterscheiden zu können.

- Für jedes Pixel ist diejenige Aufnahme zu ermitteln, bei der das Pixel am schärfsten abgebildet wird. Dazu sind die Maße aus Abschnitt **??** zu benutzen, wobei für jedes Pixel eine Fensterumgebung zu verwenden ist, die die Auflösung reduziert. Das Schärfemass wird auf homogene Flächen nicht reagieren, deshalb muss die Oberfläche gut texturiert sein.
- Erfolgen die Aufnahmen durch Änderung der Optik und nicht durch Verschiebung des Objektes ist zu beachten, dass sich dadurch Größenänderungen im Bild und Verschiebungen der beobachteten 3D-Punkte ergeben können.
- Das Verfahren bedingt einen hohen technischen und zeitlichen Aufwand.

Ein großer Vorteil ist, das die die 3D-Information ausschließlich mit einer Kamera gewonnen wird. Es sind keine Punktreferenzen notwendig und die Tiefe wird sogar absolut berechnet. Das Anwendungsspektrum ist folglich beschränkt, oft findet man im Mikroskopiebereich Anwendungen, da hier eine Höhen-Verschiebung des Objektes auf dem Mikroskoptisch normal ist und relativ leicht angewendet werden kann.

20.2. Shape from defocus

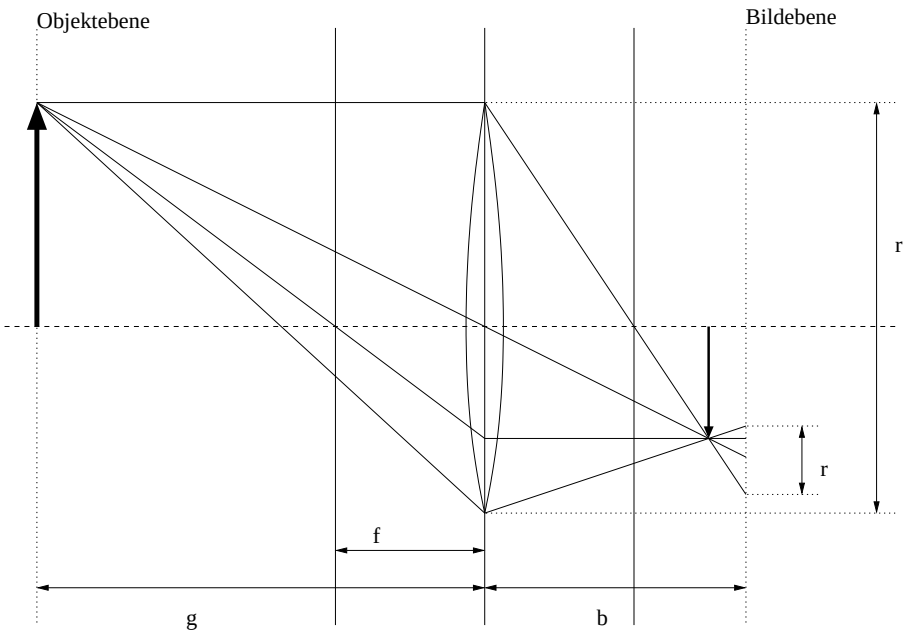


Abbildung 20.2.: Defokus - PSF einer dünnen Linse

Natürlich sind die Begriffe focus und defocus "dual" zueinander. Im folgenden soll wieder die Grundidee gezeigt werden, ausführliche Darlegungen sind z.B. in [8] zu finden. Wenn wir in Abb. 20.1 die Bildebene verschieben, aber die Objektebene belassen, dann wird der Objektpunkt "verschmiert" abgebildet. Die Größe des "verschmierten" Bereiches kann eindeutig der Entfernung des Objektpunktes zugeordnet werden. Dual zur Idee "shape from focus", wo wir die schärfste Abbildung suchen, wollen wir die gegebene Unschärfe nutzen, um daraus direkt die Tiefe des Objektpunktes zu berechnen. Wir benötigen dazu eine der Formel (20.1) entsprechende Beziehung. Diese Beziehung muss den Zusammenhang zwischen Kameraparametern, Objektpunktentfernung, Blendengröße und Größe des Verschmierungsbereiches (PSF) herstellen. Dazu nehmen wir einmal an, die Blende sei kreisförmig. Dann kann man mit der geometrischen Optik zeigen, dass der Verschmierungsbereich (PSF) auch eine Kreisfläche mit einem Helligkeitswert darstellt, oft als "pillbox" (Pillendose) bezeichnet. In der Praxis ist dies aber viel komplizierter. Eine bessere Näherung ist eine rotationssymmetrische Gaussfunktion, siehe (**??**), wobei σ die Rolle des Kreisradius der pillbox übernimmt. Es sei r der Radius der kreisförmigen Apertur, b wieder der Abstand der diesmal nicht notwendig fokussierten Bildebene zum Projektionszentrum, und r^{PSF} der Radius des "Verschmierungskreises". dann gilt

$$r^{PSF} \;=\; r \cdot f \left(\frac{1}{f} - \frac{1}{b} - \frac{1}{g} \right)$$

Benutzen wir die Gaußfunktion (**??**), dann ist $\sigma = \rho \cdot r^{PSF}$, wobei ρ ein kameraabhängiger Faktor ist, der von der Optik und der Auflösung der Sensoren abhängt. Dieser Faktor ρ sei a priori durch eine geeignete Kalibrierung bekannt. Dann ist

$$\sigma \;=\; \rho \cdot r \cdot f \left(\frac{1}{f_L} - \frac{1}{f} - \frac{1}{d_{Objekt}} \right) \quad . \tag{20.2}$$

Wenn man außer den Kameraparametern auch σ für jeden einzelnen Objektpunkt bestimmen kann, dann kann man aus (20.2) für jeden Objektpunkt die Tiefe d_{Objekt} berechnen. Wie kann man nun für jeden Objektpunkt σ berechnen?
Dazu benötigen wir mindestens zwei Bilder mit unterschiedlichen Schärfeeinstellungen, damit auch unterschiedlichen Kameraparametern, daher schreiben wir (20.2) für zwei Bilder auf:

$$\sigma_i \;=\; \rho \cdot r_i \cdot f_i \left(\frac{1}{f_{L_i}} - \frac{1}{f_i} - \frac{1}{d_{Objekt}} \right) \quad i = 1, 2 \; . \tag{20.3}$$

Wir müssen zunächst auf jeden Fall die Unbekannte d_{Objekt} eliminieren, dies gelingt durch simple Differenzbildung, wobei wir eine Gleichung erhalten:

$$\sigma_1 \;=\; a \cdot \sigma_2 \;+\; b \tag{20.4}$$

mit den Größen

$$a \;=\; \frac{r_1 f_1}{r_2 f_2} \quad , \quad b \;=\; \rho \cdot r_1 \cdot f_1 \left(\frac{1}{f_{L_1}} - \frac{1}{f_1} - \frac{1}{f_{L_2}} + \frac{1}{f_2} \right) \; .$$

Eine Gleichung drückt nun das Verhältnis der beiden "Verschmierungskreise" aus. Wir benötigen folglich noch eine zweite Beziehung, um die Größen σ_1 und σ_2 berechnen zu können. Da wir einem Objektpunkt eine PSF zuordnen, ist die PSF nicht verschiebungsinvariant, wir haben einen klassischen Fall von linearen Systemen, die nicht verschiebungsinvariant sind. Deshalb weichen wir unsere Annhamen etwas auf und nehmen an, in einer gewissen Fensterumgebung des Objektpunktes sei die Tiefe des Objekts näherungsweise konstant. Daher können wir in dieser Fensterumgebung des Punktes (x, y) Verschiebungsinvarianz annehmen und die Bildaufnahme als Faltung formulieren:

$$g_i(x, y) \;=\; h_i(x, y) \;*\; f(x, y) \quad i = 1, 2 \;. \tag{20.5}$$

Die beiden PSF $h_i(x, y), i = 1, 2$ sind die beiden modellierten Gaußfunktionen nach (**??**). Da in beide Gleichungen die gleiche Originalfunktion $f(x, y)$ eingeht, müssen wir diese wieder eliminieren, allerdings geht dies nicht durch simple Differenzbildung. Daher wenden wir auf (20.5) das Faltungstheorem (**??**) an und bilden die Quotienten aus den jeweils linken bzw. rechten Seiten der Fouriertransformierten (siehe auch (7.14)

$$\frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)} \;=\; \frac{\alpha_{h_1}(\omega_1, \omega_2)}{\alpha_{h_2}(\omega_1, \omega_2)} \;=\; e^{-\frac{1}{2}(\omega_1^2+\omega_2^2)(\sigma_1^2-\sigma_2^2)} \;.$$

Wir lösen nach $\sigma_1^2 - \sigma_2^2$ auf

$$\sigma_1^2 - \sigma_2^2 \;=\; \frac{-2}{\omega_1^2 + \omega_2^2} \log \frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)} \quad .$$

Da die rechte Seite für jedes (ω_1, ω_2) konstant sein muss, dies aber in der Praxis nicht der Fall sein wird, mitteln wir die rechte Seite durch

$$C = \; \frac{1}{A} \int \int_R \frac{-2}{\omega_1^2 + \omega_2^2} \log \frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)} \quad ,$$

wobei R ein geeigneter Mittelungsbereich ist und A die Fläche dieses Bereiches R ist. Folglich erhalten wir zu (20.4) eine zweite Gleichung:

$$\sigma_1^2 - \sigma_2^2 \;=\; C \quad . \tag{20.6}$$

Beide Gleichungen lösen wir und erhalten eine quadratische Gleichung

$$(a^2 - 1)\sigma_2^2 \;+\; 2ab\sigma_2 \;+\; b^2 \;=\; C \;, \tag{20.7}$$

welche wir leicht nach σ_2 auflösen können. Nun können wir die Beziehung (20.2) nach der Objektentfernung d_{Objekt} auflösen.
Diese Berechnung müssen wir nun für jedes Pixel (x, y) durchführen und erhalten aus lediglich zwei Bildern, die mit verschiedenen Kameraparametern aufgenommen wurden, eine vollständige Tiefenkarte. Die Probleme, die bei der praktischen Realisierung auftreten, sind die folgenden:

- Ein Vorteil ist, dass die Methode sehr simpel ist, nur zwei Aufnahmen verlangt und keinerlei Punktereferenzen benötigt. Der Preis dafür ist, dass sie in der Genauigkeit etwas schlechter als die leistungsfähigen Stereo-Methoden ist. Der durchschnittliche Tiefenfehler liegt bei ca. 5 - 8 Prozent.

- Da wir die PSF durch eine Gauß-Funktion modelliert haben, entstehen Modellierungsfehler. Andere Modelle könnten besser sein.

- Die Kameraparameter müssen für die beiden Aufnahmen sehr genau bestimmt werden.

- Die größte Fehlerquelle ist das Problem der lokalen Fensterwahl. Die Annahme, in einer lokalen Fensterumgebung sei die Tiefe des Objektes annähernd konstant, d.h. wir können lokal die Faltung mit der PSF nutzen, ist sehr schwer zu überprüfen. Was passiert z.B., wenn sich die Tiefe abrupt ändert?

- Wenn die Grauwertfunktion $f(x, y)$ über einem lokalen Bereich konstant ist, dann bedeutet eine Faltung lediglich eine Multiplikation mit einem Faktor. Man kann folglich die relative Änderung der "Verschmierung" zwischen zwei Aufnahmen nicht bestimmen und das Verfahren funktioniert nicht. Die Oberfläche muss folglich wie bei "shape from focus" gut texturiert sein.

Teil IX.

Momente als signalbasierte Merkmale

21. Momente

21.1. Momente

21.1.1. Flächenmomente

Die Flächenmomente der Ordnung $p+q$ für die Grauwertfunktion $f(x,y)$ eines Bildes sind definiert zu:

$$m_{p,q} = \iint_{DB} x^p y^q f(x,y)\ dx\ dy \tag{21.1}$$

Integriert wird über den gesamten Definitionsbereich DB. Dabei ist die Funktion $f(x,y)$ eine Gewichtung, die eine Bewertung des Punktes x,y darstellt. Eine spezielle Gewichtung ist die Entscheidung, ob ein Punkt überhaupt berücksichtigt werden muss. Das kann die Zugehörigkeit zu einem Objekt sein. In diesem Fall wäre $f(x,y)$ gleich 1 für Objektpunkte, Null für Punkte außerhalb. Da Punkte außerhalb damit nicht zum Integral beitragen, kann die Momentberechnung auch als

$$m_{p,q} = \iint_{B} x^p y^q\ dx\ dy \tag{21.2}$$

geschrieben werden, wobei B die Objektfläche bezeichnet.
Einige Momente lassen eine unmittelbare Interpretation zu:

- $A = m_{0,0}$ ist die Fläche von B, falls $f(x,y) = 1$.
- $x_c = \frac{m_{1,0}}{m_{0,0}}$ ist die x-Koordinate des Schwerpunktes von B.
- $y_c = \frac{m_{0,1}}{m_{0,0}}$ ist die y-Koordinate des Schwerpunktes von B.

Die Momente 2. Ordnung beschreiben die sogenannte Trägheitsellipse:

$$m_{0,2} \cdot x^2 - 2 \cdot m_{1,1} \cdot x \cdot y + m_{2,0} \cdot y^2 = 1 \tag{21.3}$$

Für eine sinnvolle Interpretation der Trägheitsellipse sollte der Koordinatenursprung in den Schwerpunkt gelegt werden. Nun kann man die Orientierung dieser Ellipse, den Winkel zwischen der Hauptachse der Ellipse und der x-Achse, als Orientierung des Objektes B benutzen. Dieser Winkel ergibt sich zu

$$\varphi = \frac{1}{2} \arctan \frac{2m_{1,1}}{m_{0,2} - m_{2,0}} \tag{21.4}$$

Wenn nun die Trägheitsellipse zum Trägheitskreis entartet oder näherungsweise einen Trägheitskreis darstellt, dann kann man die Orientierung nicht mehr mit den Momenten 2. Ordnung beschreiben. Zähler und Nenner in Gleichung 21.4 werden gleichzeitig Null oder näherungsweise Null. In diesem Falle kann man die Momente 3. bis 4. Ordnung zur Orientierung des Objektes benutzen, siehe [38].

21.1.2. Linienmomente und Punktmomente

Die Linienmomente sind analog zu den Flächenmomenten definiert, sie beziehen sich aber auf ein Liniensegment L:

$$m_{p,q} = \int_L x^p y^q f(x,y) ds \tag{21.5}$$

Im Gegensatz zum Flächenintegral 21.1 wird in 21.5 ein Kurvenintegral 1. Ordnung bezüglich der Bogenlänge s benutzt.
Die Punktmomente beziehen sich auf eine endliche, diskrete Punktmenge P:

$$m_{p,q} = \sum_{(x_i,y_i)\in P} x_i^p y_i^q f(x_i,y_i) \tag{21.6}$$

Am häufigsten werden in der Bildverarbeitung Flächenmomente genutzt, da deren Ermittlung robust möglich ist. Linienmomente von Objektkanten oder ähnlichem werden stark durch Rauschen gestört. Punktmomente erfordern eine stabile Ermittlung der Punkte, deren Anzahl beziehungsweise Dichte konstant bleiben muss, um Vergleichbarkeit zu ermöglichen.

21.1.3. Komplexe Momente

Betrachtet man die Ebene als Gaußsche Zahlenebene, kann man jeden Punkt als komplexe Zahl $z = x + iy$ auffassen und damit komplexe Momente definieren:

$$C_{p,q} = \iint_B z^p\ \overline{z}^q f(z)\ dx\ dy = \iint_B (x+iy)^p (x-iy)^q f(x,y)\ dx\ dy \tag{21.7}$$

Man sieht, dass man die komplexen Momente durch "auspotenzieren" auf die gewöhnlichen Flächenmomente zurückführen kann. Die komplexe Schreibweise kann manchmal die Darstellung und Rechnung vereinfachen. Linien- und Punktmomente lassen sich analog als komplexe Momente formulieren.

21.2. Momente und Fouriertransformation

Betrachten wir die Fourier-Integraltransformation IFT (6.5) für zwei unabhängige Variablen und entwickeln den e-Term in eine Potenzreihe:

$$\begin{aligned}\alpha(\nu_1,\nu_2) &= \iint\limits_B f(x,y)e^{-2\pi i(\nu_1 x+\nu_2 y)}\,dx\,dy \\ &= \iint\limits_B f(x,y)\sum_{j=0}^{\infty}\frac{(-2\pi i\nu_1 x)^j}{j!}\sum_{k=0}^{\infty}\frac{(-2\pi i\nu_2 y)^k}{k!}\,dx\,dy\end{aligned}$$

Vertauschen von Integration und Summation liefert

$$\begin{aligned}\alpha(\nu_1,\nu_2) &= \sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\left(\frac{(-2\pi i)^{j+k}}{j!k!}\iint x^j y^k\,dx\,dy\right)\cdot {\nu_1}^j{\nu_2}^k \\ &= \sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\left(\frac{(-2\pi i)^{j+k}}{j!k!}m_{j,k}\right)\cdot {\nu_1}^j{\nu_2}^k\end{aligned}$$

Wir haben das Spektrum $\alpha(\nu_1,\nu_2)$ in eine Potenzreihe entwickelt, wobei die Koeffizienten bis auf einen Faktor die Momente der Funktion $f(x,y)$ sind. Damit wird das Spektrum vollständig durch die Momente beschrieben. Da das Spektrum aber vollständig die Originalfunktion $f(x,y)$ beschreibt, haben wir gezeigt, dass auch die Momente nur eine andere Darstellung der Funktion sind.
Auch formal ist die Definition der Flächenmomente in Gleichung 21.1 einer verallgemeinerten Fourier-Transformation 5.2 sehr ähnlich. Allerdings ist die Rücktransformation nicht ganz so einfach zu ermitteln, da die $x^k y^j$ keine orthonormalen Funktionen sind. Mit entsprechend orthonormalisierten Basisfunktionen kann man dazu besser geeignete Momente definieren. Die *Legendre Momente* und die *Zernike Momente* sind Beispiele dafür ([28]).

21.3. Transformation der Momente

Wenn wir ein Bild/Objekt B transformieren, dann ändern sich natürlich auch die Momente. Diese kann man vom transformierten Bild B' berechnen. Man kann aber auch versuchen, die Auswirkung der Transformation auf die Momente anzugeben und die Momente direkt transformieren.
Die einfachste Transformation ist die Translation (13.2).

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix} \tag{21.8}$$

Wir wollen zunächst die Flächenmomente transformieren. In die Berechnung der transformierten Momente setzen wir die transformierten Koordinaten ein. Bei Flächenintegralen ändern sich die Differentiale mit dem Betrag der Funktionaldeterminante. Diese ist bei Translationen Eins. Folglich gilt

$$m'_{p,q} = \iint\limits_{B'} x'^p y'^q f'(x',y')\,dx'\,dy' = \iint\limits_{B}(x+a_{10})^p(y+a_{20})^q f(x,y)\,dx\,dy$$

Für die ersten Momente ergibt sich konkret

$$\begin{aligned} m'_{0,0} &= m_{0,0} \\ m'_{1,0} &= m_{1,0} + a_{10} m_{0,0} \\ m'_{0,1} &= m_{0,1} + a_{01} m_{0,0} \end{aligned}$$

Analog kann man für beliebige Transformationen die Momente umrechnen. Bei affinen Transformationen ist zum Beispiel die Funktionaldeterminante gleich $det(A) = a_{11}a_{22} - a_{12}a_{21}$. Das Moment $m_{0,0}$ transformiert sich deshalb bei Flächenmomenten nach

$$m'_{0,0} = det(A) m_{0,0} \tag{21.9}$$

Die Transformationen der Linien- und Punktmomente weichen davon ab. Für die Momente m_{00} ergibt sich beispielsweise bei einer Ähnlichkeitstransformation mit einer Skalierung von s:

Flächenmomente	$m'_{0,0} = s^2 \; m_{0,0}$
Linienmomente	$m'_{0,0} = s \; m_{0,0}$
Punktmomente	$m'_{0,0} = s^0 \; m_{0,0} = m_{0,0}$

Im folgenden sollen vor allem die Flächenmomente weiter betrachtet werden, da diese als Merkmale für Objekte eine große Rolle spielen.

21.4. Normalisierung der Momente

Die Idee der Normalisierung ist zunächst sehr allgemein: Existieren mehrere verschiedene Beschreibungen eines Objektes, so wählt man davon eine aus, die dann mit anderen vergleichbar ist. Als sehr einfaches Beispiel seien verschiedene Darstellungen einer Zeit genannt: 100 min = 1 h 40 min = $1.\overline{6}$ h . Will man jetzt zwei Zeiten 3 h 20 min und 200 min miteinander vergleichen, so ist dies nicht offensichtlich. Normalisiert man die Angaben auf die übliche Darstellung in Stunden und Minuten sieht man, dass beide mit 3 h 20 min gleich sind.
Bei der Benutzung von Momenten als Merkmalen für Objekte ergeben sich unterschiedliche "Darstellungen" durch die unterschiedliche Lage und Größe der Objekte. Oft wird man Objekte auch dann als gleich bezeichnen, wenn sie sich durch eine Transformation (Verschiebung, Rotation, Skalierung oder ähnliches) unterscheiden. Die Merkmale sollten sich dann nicht unterscheiden, also invariant bezüglich der Transformation sein. Die Benutzung der Normalisierung zur Bestimmung invarianter Merkmale bedeutet somit, eine Transformation zu finden, die Objekte in eine Standardlage ("canonical frame") überführt, so dass dann die Merkmale gleich sind.
Hat man sich auf eine Transformation T und eine Art von Merkmalen m_i festgelegt, kann man die Merkmale durch Normalisierung invariant machen:

- Man bestimmt, wie die Merkmale durch T transformiert werden: $m'_i = T(m_i)$

- Man legt für bestimmte Merkmale m_k geeignete Werte fest, die die Standardlage beschreiben: $m'_k = M_k$
- Man ermittelt eine Transformation T_n, die die Merkmale in die für die Standardlage definierten Merkmale überführt: $m'_k = T_n(m_k) = M_k$
- Man transformiert alle Merkmale (und eventuell das Objekt) in die Standardlage: $m'_i = T_n(m_i)$

Damit erhält man invariante Merkmale m' und eine Transformation T_n. Man kann somit sowohl die invarianten Merkmale für den Vergleich zweier Objekte nutzen, als auch die Objekte transformieren und diese direkt oder mittels anderer Merkmale vergleichen.

21.4.1. Normalisierung der Momente bezüglich Translation

Transformation: Translation (13.2)

$$\begin{array}{rcl} x' & = & x + \Delta x \\ y' & = & y + \Delta y \end{array} \tag{21.10}$$

Transformation der Momente:

$$m'_{p,q} = \iint\limits_{B'} x'^p y'^q f'(x', y')\ dx'\ dy' = \iint\limits_{B} (x + \Delta x)^p (y + \Delta y)^q f(x, y)\ dx\ dy$$

Standardlage:

$$m'_{1,0} = 0,\ m'_{0,1} = 0$$

Bestimmung der Transformation:

$$\begin{array}{ll} m'_{1,0} = m_{1,0} + \Delta x\ m_{0,0} = 0 & m'_{0,1} = m_{0,1} + \Delta y\ m_{0,0} = 0 \\ \rightarrow \Delta x = -\dfrac{m_{1,0}}{m_{0,0}} = x_c & \rightarrow \Delta y = -\dfrac{m_{0,1}}{m_{0,0}} = y_c \end{array}$$

Normalisierte Momente:

$$m'_{p,q} = \iint\limits_{B} (x - \frac{m_{1,0}}{m_{0,0}})^p (y - \frac{m_{0,1}}{m_{0,0}})^q f(x, y)\ dx\ dy$$

Die so bestimmte Translation verschiebt den Schwerpunkt (x_c, y_c) des Objekts in den Koordinatenursprung. Die danach berechneten Momente sind *invariant* bezüglich Translation. Diese Momente heißen *zentrale Momente* und werden oft mit $\mu_{p,q}$ bezeichnet. Wir haben hier auf eine explizite Darstellung der Transformation der Momente $m'_{p,q} = T(m_{p,q})$ verzichtet, da die Formeln nach Ausmultiplizieren der Potenzen umfangreich und unübersichtlich sind. Bei einer Implementierung des Verfahrens muss man diese Ausdrücke für die Momente bis zu einer gegebenen Ordnung (zum Beispiel 5) umsetzen.

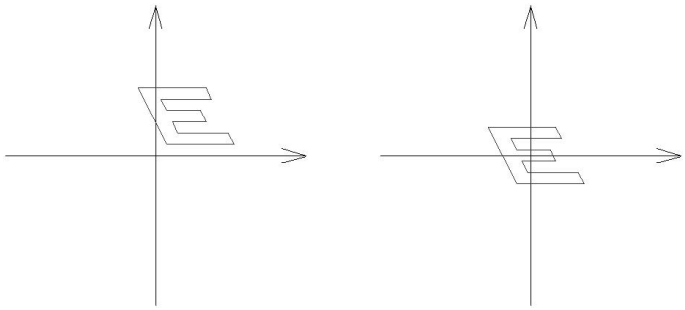


Abbildung 21.1.
Original-Objekt und bezüglich Translation normalisiertes Objekt

21.4.2. Normalisierung der Momente bezüglich X-Scherung

Transformation: X-Scherung (13.7)

$$\begin{array}{rcl} x' & = & x + s_x y \\ y' & = & y \end{array} \tag{21.11}$$

Transformation der Momente:

$$m'_{p,q} = \iint\limits_{B'} x'^p y'^q f'(x', y')\ dx'\ dy' = \iint\limits_{B} (x + s_x y)^p (y)^q f(x, y)\ dx\ dy$$

$$m'_{1,1} = m_{1,1} + s_x m_{0,2}$$

Standardlage:

$$m'_{1,1} = 0$$

Bestimmung der Transformation:

$$m_{1,1} + s_x m_{0,2} = 0$$

$$s_x = -\frac{m_{1,1}}{m_{0,2}}$$

Normalisierte Momente

$$m'_{p,q} = \iint\limits_{B} (x - \frac{m_{1,1}}{m_{0,2}} y)^p (y)^q f(x, y)\ dx\ dy$$

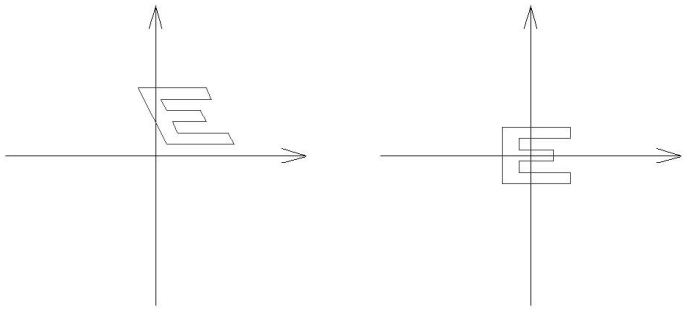


Abbildung 21.2.
Original-Objekt und bezüglich Translation und X-Scherung normalisiertes Objekt

21.4.3. Normalisierung der Momente bezüglich anisotroper Skalierung

Nach der Normalisierung bezüglich Translation und X-Scherung liegt der Schwerpunkt des Objekts im Koordinatenursprung und die Achsen der Trägheitsellipse liegen auf den Koordinatenachsen. Durch eine Normierung bezüglich der anisotropen Skalierung (13.10) wird aus der Trägheitsellipse ein Trägheitskreis.
Transformation: anisotrope Skalierung

$$\begin{aligned} x' &= cx \\ y' &= dy \end{aligned} \tag{21.12}$$

Transformation der Momente:

$$m'_{p,q} = \iint\limits_{B'} x'^p y'^q f'(x', y')\ dx'\ dy' = \iint\limits_{B} (cx)^p (dy)^q f(x, y)\ dx\ dy$$

$$\begin{aligned} m'_{2,0} &= c^3 d m_{2,0} \\ m'_{0,2} &= c d^3 m_{0,2} \end{aligned}$$

Standardlage:

$$m'_{2,0} = 1,\ m'_{0,2} = 1$$

Bestimmung der Transformation:

$$\begin{aligned} c &= \sqrt[8]{\frac{m'^3_{0,2}}{m'^3_{2,0}}} \\ d &= \sqrt[8]{\frac{m'^3_{2,0}}{m'^3_{0,2}}} \end{aligned}$$

Transformation der Momente:

$$m'_{p,q} = c^p d^q m_{p,q}$$

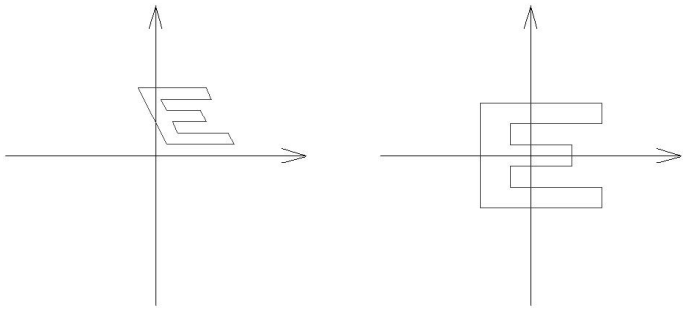


Abbildung 21.3.
Original-Objekt und bezüglich Translation, X-Scherung und Skalierung normalisiertes Objekt

21.4.4. Normalisierung der Momente bezüglich Rotation

Transformation: "reine" Rotation

$$\begin{array}{rcl} x' & = & \cos\varphi \cdot x \quad - \quad \sin\varphi \cdot y \\ y' & = & \sin\varphi \cdot x \quad + \quad \cos\varphi \cdot y \end{array} \tag{21.13}$$

Transformation der Momente:

$$m'_{p,q} = \iint\limits_{B'} x'^p y'^q f'(x', y')\, dx'\, dy' = \iint\limits_{B} (\cos\varphi \cdot x - \sin\varphi \cdot y)^p (\sin\varphi \cdot x + \cos\varphi \cdot y)^q f(x, y)\, dx\, dy$$

$$\begin{aligned} m'_{1,0} &= \cos\varphi \cdot m_{1,0} - \sin\varphi \cdot m_{0,1} \\ m'_{0,1} &= \sin\varphi \cdot m_{1,0} + \cos\varphi \cdot m_{0,1} \end{aligned}$$

Standardlage:

$$m'_{1,0} = 0$$

Bestimmung der Transformation:

$$\varphi_n = \arctan \frac{m_{1,0}}{m_{0,1}} + k \cdot \pi \quad \text{mit} \quad k \in \mathbb{Z}$$

Transformation der Momente:

$$m'_{p,q} = \iint\limits_{B} (\cos\varphi \cdot x - \sin\varphi \cdot y)^p (\sin\varphi \cdot x + \cos\varphi \cdot y)^q f(x, y)\, dx\, dy$$

Wir haben, wie auch bei den Normalisierungsbeispielen zuvor, die Transformation der Momente in die Standardlage nicht ausgeschrieben. Das Ausmultiplizieren der Potenzen liefert umfangreiche Ausdrücke, die nicht mehr anschaulich sind. Für die Ausführung der Normalisierung muss man in der sauren Apfel beißen und diese Ausdrücke bis zu einer gewissen Ordnung berechnen.
Fast immer wird man zuvor bezüglich Translationen normalisieren. Dann sind aber die Momente $m_{1,0}$ und $m_{0,1}$ bereits Null und zur Normalisierung der Rotation nicht mehr verwendbar. Aber es lassen sich Momente höherer Ordnung nutzen, die noch nicht für andere Normalisierungen benutzt wurden.
Aber es gibt noch ein weiteres Problem: Welches Moment man auch für die Normalisierung auswählt - es gibt Objekte mit Symmetrien, bei denen dieses Moment bei Rotation konstant ist und der Drehwinkel nicht bestimmt werden kann. Für jede Art von speziellen Objekten kann man geeignete Normalisierungen finden, allgemeingültige für alle Objekte nicht.
Ist das Ziel der Normalisierung, invariante Merkmale zu finden, kann man aber einen alternativen Weg gehen: Man verwendet Ausdrücke mit Momenten, die invariant gegen Rotation sind. Das trivialste Beispiel ist $m_{0,0}$, welches nicht von der Rotation abhängt. Bekannte Lösungen sind die Hu-Invarianten, beschrieben in Abschnitt 21.5.1.

21.4.5. Kombinationen

Normalisierungen lassen sich kombinieren. Dabei ist natürlich zu beachten, dass eine nachfolgende Normalisierung kein Moment verwenden darf, welches durch eine vorhergehende Normalisierung bereits konstant ist. Auch darf eine nachfolgenden Normalisierung nicht eine Moment verändern, welches bereits vorher auf einen festen Wert gesetzt wurde. Eine sinnvolle Reihenfolge ist

- Normalisierung der Translation (21.4.1)
- Normalisierung der X-Scherung (21.4.2)
- Normalisierung der anisotropen Skalierung (21.4.3)
- Normalisierung der Rotation mittels der Momente 3. Ordnung (21.4.4)

Diese Kombination liefert eine Normalisierung bezüglich affiner Transformationen (13.11). Je nach Aufgabenstellung kann es sinnvoll sein, Schritte in dieser Kombination auszulassen.

- Eine Anwendung zur Texterkennung (OCR) verzichtet auf die Normalisierung der Rotation, da die Orientierung typischerweise fest ist und Information trägt, wie zum Beispiel bei der Unterscheidung von 6 und 9.
- Treten bei der Erfassung der Aufnahmen keine Scherungen auf, wie zum Beispiel beim Flachbett-Scanner, dann ist eine Scherung eine wichtige Information zum Objekt. Man wird hier auf die Normalisierung der Scherung verzichten.

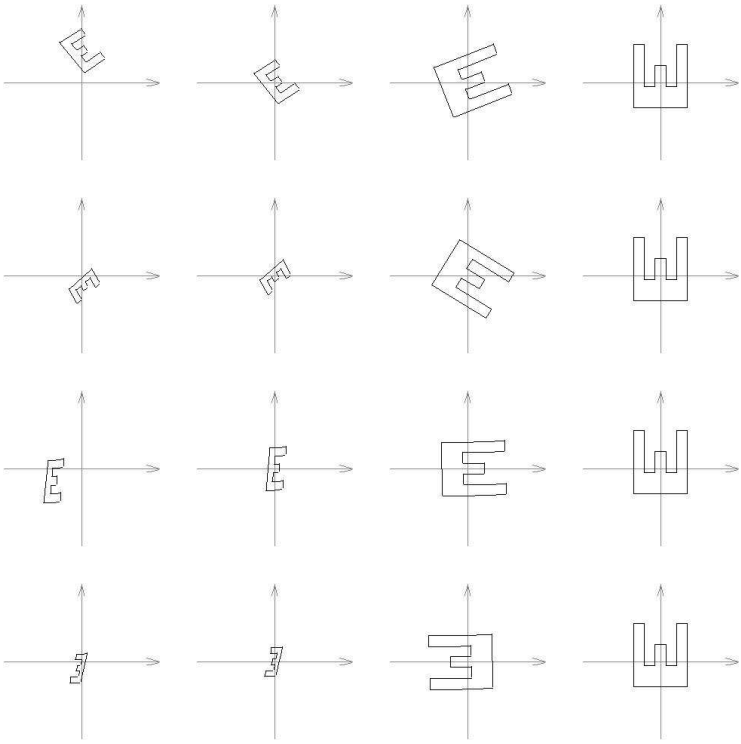


Abbildung 21.4.
Ein Objekt in verschiedenen Lagen
Original, normalisiert Translation, X-Scherung und Skalierung,
Rotation (v.l.n.r.)

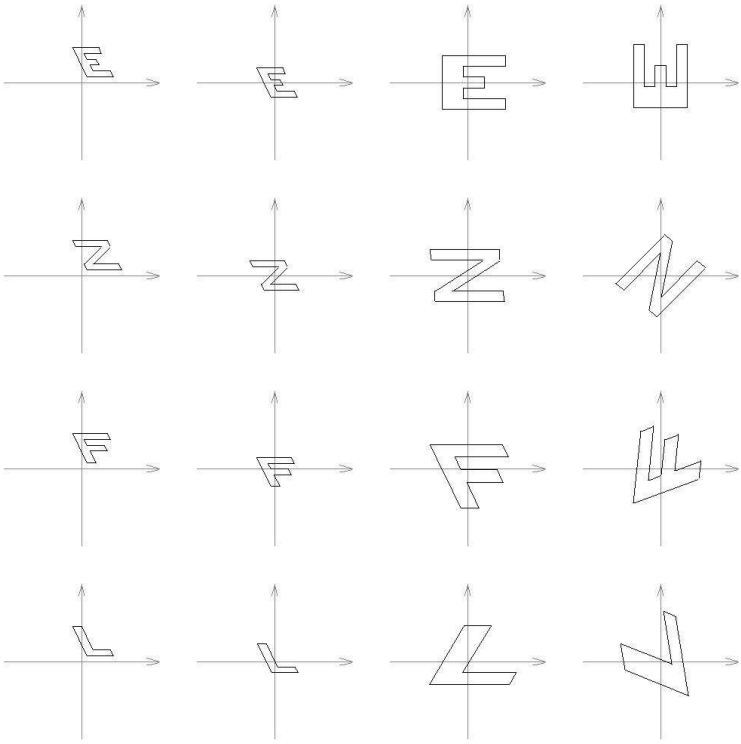


Abbildung 21.5.
Verschiedene Objekte und ihre Standardlage
Original, normalisiert Translation, X-Scherung und Skalierung,
Rotation (v.l.n.r.)

21.5. Invarianten aus Momenten

21.5.1. Invarianten für reine Rotationen

Da die Rotation im Prozess der Normalisierung schwierig zu behandeln ist (21.4.4), könnte man statt dessen Invarianten suchen, die sich aus den Momenten berechnen lassen. Dies sind keine Momente mehr und eine Transformation in eine Standardlage erhält man so nicht. 7 Invarianten für reine Rotationen hat Hu ([18]) bereits 1962 gefunden. Herbert Süße hat eine allgemeine Methode zur Bestimmung solcher Invarianten auf der Basis der komplexen Momente gefunden und die Liste dieser Invarianten erweitert ([39]). In der folgenden Liste entspricht die Numerierung 1 bis 7 der Hu-Momente der Veröffentlichung von Hu, die zusätzlichen Invarianten nach Süße sind entsprechend eingeordnet:

$$
\begin{aligned}
H_{-1} &= m_{0,0} \\
H_0 &= m_{1,0}^2 + m_{0,1}^2 \\
H_1 &= m_{2,0} + m_{0,2} \\
H_2 &= (m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2 \\
H_3 &= (m_{3,0} - 3m_{1,2})^2 + (3m_{2,1} - m_{0,3})^2 \\
H_4 &= (m_{3,0} + m_{1,2})^2 + (m_{2,1} + m_{0,3})^2 \\
H_5 &= (m_{3,0} - 3m_{1,2}) \left[(m_{3,0} + m_{1,2})^3 - 3(m_{2,1} + m_{0,3})^2(m_{3,0} + m_{1,2})\right] \\
&\quad +(m_{0,3} - 3m_{2,1}) \left[(m_{0,3} + m_{2,1})^3 - 3(m_{1,2} + m_{3,0})^2(m_{0,3} + m_{2,1})\right] \\
H_6 &= (m_{3,0} - 3m_{1,2}) \left[(m_{0,3} + m_{2,1})^3 - 3(m_{1,2} + m_{3,0})^2(m_{0,3} + m_{2,1})\right] \\
&\quad -(m_{0,3} - 3m_{2,1}) \left[(m_{3,0} + m_{1,2})^3 - 3(m_{2,1} + m_{0,3})^2(m_{3,0} + m_{1,2})\right] \\
H_7 &= (m_{2,0} - m_{0,2}) \left[(m_{3,0} + m_{1,2})^2 - (m_{2,1} + m_{0,3})^2\right] \\
&\quad +4m_{1,1} \left[(m_{3,0} + m_{1,2})(m_{2,1} + m_{0,3})\right] \\
H_8 &= (m_{4,0} - m_{0,4})^2 + 4(m_{3,1} + m_{1,3})^2 \\
H_9 &= (m_{4,0} + m_{0,4})^2 + 16(m_{3,1} - m_{1,3})^2 - 12m_{2,2}(m_{0,4} - 3m_{2,2} + m_{0,4}) \\
H_{10} &= (m_{4,0} - m_{0,4})(m_{2,0} - m_{0,2}) + 4m_{1,1}(m_{3,1} + m_{1,3}) \\
H_{11} &= m_{4,0} + 2m_{2,2} + m_{0,4} \\
H_{12} &= (m_{4,0} - 6m_{2,2} + m_{0,4}) \left[(m_{2,0} - m_{0,2})^2 - 4m_{1,1}^2\right] \\
&\quad -16 \left[m_{1,1}(m_{2,0} - m_{0,2})(m_{3,1} - m_{1,3})\right]
\end{aligned}
\tag{21.14}
$$

21.5.2. Affin-invariante Merkmale

Die Invarianten in 21.14 sind invariant bezüglich reiner Rotationen. Durch eine vorhergehende Normalisierung erreicht man Invarianz gegen weitere Transformationen:

- Normalisierung der Translation (21.4.1)
- Normalisierung der X-Scherung (21.4.2)
- Normalisierung der anisotropen Skalierung (21.4.3)
- Berechnung der Invarianten H bezüglich der Rotation nach (21.14)

Diese Kombination erzeugt affin-invariante Objekt-Merkmale. Diese sind robuster als die reine Normalisierung (21.5.2), bestimmen aber keine Transformation in die Standardlage. Man muss beachten, dass einige der Invarianten in 21.14 nicht mehr nutzbar sind, wenn die Momente durch die Normalisierung konstant gesetzt wurden. Auch bei der Berechnung der Invarianten kann man beliebige Zwischenschritte auslassen, wenn eine bestimmte Invarianz nicht gewünscht wird (siehe auch 21.4.5).

22. Transformations-Bestimmung aus Punktmengen

Die Bestimmung von Transformationen aus der Kenntnis von Referenzpunktpaaren ist keine Aufgabe der Signaltheorie. Hat man ein parametrisiertes Modell der Transformation, so kann man aus der Kenntnis der Punktreferenzen die Parameter der Transformation berechnen. Bei der affinen Transformation (13.11) führt dies zu einem linearen Gleichungssystem:

$$\mathbf{x'}_i = \mathbf{A}\mathbf{x}_i + \mathbf{a}_0 \quad \text{mit} \quad i = 0..N$$

Dies lässt sich nun zum Beispiel mit den Gaußschen Normalengleichungen lösen und die 6 Unbekannten der affinen Transformation bestimmen.
Methoden der Signaltheorie können nützlich sein, wenn die Referenzen ermittelt werden müssen. Ausgangspunkt können Punktmengen sein, deren Zuordnung noch unbekannt ist. Je nach Aufgabenstellung können die Punktmengen bereits mehr oder weniger geordnet sein.

22.1. Geordnete Mengen von Punkten

Sind zwei Punktmengen gegeben, wobei die zweite die transformierten Punkte der ersten Menge enthält, so ist ein erster Schritt zur Bestimmung der Transformation die Zuordnung der Punkte. Wir wollen jetzt Punktmengen betrachten, die eine Ordnung besitzen. Beispielhaft seien die folgenden Fälle genannt:

- Gegeben sind die Eckpunkte eines Polygons. Die Punkte besitzen eine Ordnung im Sinne des Umlaufs um das Polygon. Die Zahl der Punkte in beiden Mengen ist im Idealfall gleich. Es bleibt die Aufgabe, die Position des ersten Punktes der ersten Menge in der zweiten Menge zu finden. Alle anderen Punktzuordnungen sind dann festgelegt.

- Gegeben ist ein Binärobjekt B. Mittels eines Konturfolgeverfahrens wird die Kontur des Objektes bestimmt. Diese Kontur ist ebenfalls eine geordnete Menge von Punkten. Die Konturen zweier Objekte unterscheiden sich wie bei Polygonen im Startpunkt, aber durch Skalierung und Quantisierungseffekte kann sich auch die Zahl der Punkte auf der Kontur unterscheiden.

- Einem Bild g ordnen wir *affin kovariant* eine geordnete Folge von Punkten zu. Wir bestimmen zunächst den Schwerpunkt x_c des Bildes. Wir betrachten eine Gerade

durch x_c und berechnen den Schwerpunkt aller Grauwerte "rechts" von dieser Geraden. Wir drehen die Gerade um x_c und berechnen für jeden Winkel den Schwerpunkt "rechts" der Geraden. Wir erhalten mit der Folge von Schwerpunkten eine geordnete Folge von Punkten.
Damit diese Punktfolge wirklich aussagekräftig ist, sollten die Grauwerte sollten zum Bildrand hin abfallen und in einem gewissen Abstand zum Rand näherungsweise verschwinden. Damit bestimmt der Bildinhalt die Punktfolge und nicht der Rand des Bildes. Ein solches Bild entsteht beispielsweise durch Bilden des Amplituden- oder Leistungsspektrums eines Bildes.

Im folgenden soll eine äußerst robuste Registrierungsmethode für eine solche Situation beschrieben werden ([37]). Wir benötigen dazu

- Merkmale, die die Punkte unterscheiden und eine Zuordnung ermöglichen
- einen Algorithmus, der die Punkte zuordnet und dabei die vorhandene Ordnung ausnutzt.

22.1.1. Affin-invariante Punkt-Merkmale

Für jeden Punkt berechnen wir Merkmale folgenderweise:

- Wir legen in den jeweiligen Punkt den Koordinatenursprung.
- Wir legen eine affin-kovariante Umgebung U des Punktes fest. Für affine Transformationen ist dies schwierig, weshalb wir als Umgebung das gesamte Bild oder bei Binärobjekten das gesamte Objekt wählen.
- Bezüglich des gewählten Koordinatenursprungs und der Umgebung U berechnen wir jetzt die affinen Momentinvarianten $H_{-1}, H_0, H_1, \cdots, H_{12}$ aus Abschnitt 21.5.2. Wichtig hier: Wir dürfen nicht gegenüber der Translation normalisieren (zentrale Momente), sondern die Momente sind bezüglich des festgelegten Koordinatenursprungs zu berechnen. Die Invarianten sind nun invariant gegen den linearen Anteil A einer affinen Transformation. Wir können auch die Invariante H_0 benutzen, da die Momente 1. Ordnung nicht normalisiert werden. Wir benutzen die nicht normalisierten Momente bis 4. Ordnung $H_0, H_3, H_4, \cdots, H_{12}$. Die Invarianten H_1 und H_2 sind nicht benutzbar, da sie durch die Normalisierung der Trägheitsellipse stets konstant sind.

Wir haben jedem Punkt aus der geordneten Punktfolge einen 11-dimensionalen Merkmalsvektor aus den Invarianten zugeordnet. Dies tun wir für die originale und die transformierte Punktfolge.
Jetzt ist das Referenzproblem zu lösen. Es sind die Punkte aus B den Punkten aus B' zuzuordnen, die die gleichen Invarianten besitzen. Dazu benutzen wir ein Abstandsmaß, zum Beispiel die Euklidische Metrik. Ordnen wir mit einer Standardmethode (z.B. Nächste Nachbarsuche, Ungarische Methode) die Punkte einander zu, verschenken wir aber eine wichtige Information: die gegebene Ordnung innerhalb der Mengen. Bei der Zuordnung der Punkte mittels dynamischer Programmierung nutzen wir die vorhanden Ordnung aus.

22.1.2. Zuordnung mittels Dynamischer Programmierung

Wir berechnen eine Kostenfunktion $d(i, j)$ zwischen dem Merkmalsvektor des i-ten Punktes aus B und dem Merkmalsvektor des j-ten Punktes aus B'. Wenn die Merkmalsvektoren ähnlich sind, dann sollte es geringe Kosten geben, und wenn sie unähnlich sind, dann sollte es große Kosten geben. Die einfachste Kostenfunktion ist der Euklidische Abstand zwischen den Merkmalsvektoren, den wir auch im folgenden benutzen. In der Abbildung 22.1 links ist eine typische Distanzmatrix zu sehen.

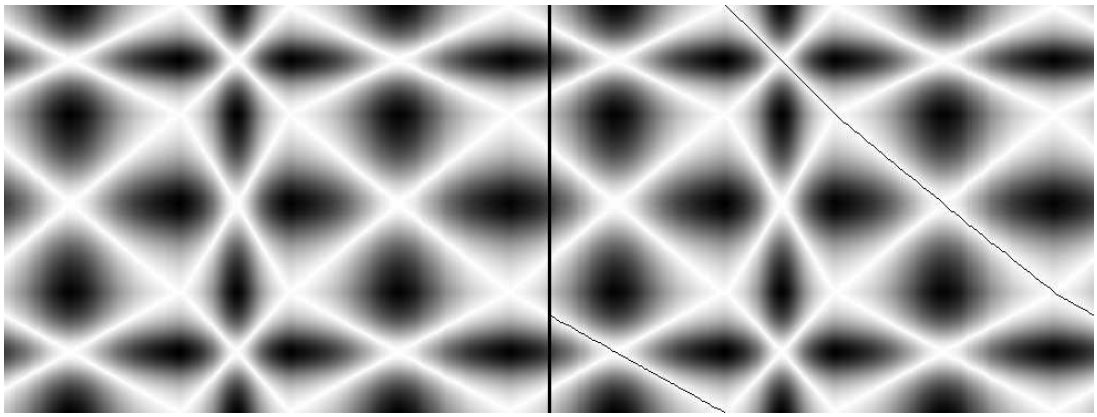


Abbildung 22.1.
Distanz-Funktion und detektierter, optimaler Pfad

Wir suchen eine Folge von Referenzpunktpaaren (i, j) mit minimalen Kosten

$$\sum_{(i,j)} d(i,j) \;\rightarrow\; Minimum$$

Dies entspricht einem Pfad von der ersten Spalte bis zur letzten Spalte mit minimaler Kostensumme. Da die Punktfolgen zyklisch sind, müssen wir einen Pfad, der den Rand der Matrix erreicht, auf der anderen Seite periodisch fortsetzen. In der Abbildung 22.1 ist rechts der optimalen Pfad mit periodischer Fortsetzung eingezeichnet.
Nun wollen wir den optimalen Pfad berechnen. Dazu nehmen wir an, dass es in horizontaler Richtung (Index i) mehr Punkte als in vertikaler Richtung (Index j) gibt, notfalls vertauschen wir einfach die Punktmengen. Jeder Pfad führt über alle Spalten i und ordnet ein j zu. Jedem Punkt der größeren Menge wird genau ein Punkt der kleineren Menge zugeordnet, zu jedem Punkt der kleineren Menge gehört eventuell mehr als ein Punkt der größeren Menge. Die Pfadberechnung erfolgt von links nach rechts. In einem Schritt wird jedem Punkt j einer Spalte als Vorgänger der Punkt j oder $j-1$ der Vorgängerspalte zugeordnet und die (minimalen) Kosten für das Erreichen dieses Punktes berechnet. In der letzten Spalte ist dann der Punkt mit minimalen Kosten der Endpunkt des optimalen Pfades und der optimale Pfad kann somit zurückverfolgt werden.

Der Algorithmus kann rekursiv formuliert werden:

Optimaler Pfad zu einem Punkt j der Spalte i und Kosten $c(i,j)$:

$i = 0$ Jeder Punkt j ist möglicher Startpunkt eines Pfades und die Kosten des Pfades sind die Kosten des Punktes $d(0,j)$.

$$c(0,j) = d(0,j)$$

$i > 0$ Der optimale Pfad ist der zu einem Vorgänger j oder $j-1$ in der Spalte $i-1$ und die Fortsetzung zum aktuellen Punkt mit den zusätzlichen Kosten $d(i,j)$. Von den beiden Vorgängern j oder $j-1$ ist derjenigen mit den geringsten Kosten c ihrer minimalen Pfade zu wählen

$$c(i,j) = min(c(i-1,j), c(i-1,j-1)) + d(i,j)$$

In der praktischen Umsetzung wird man meist iterativ (statt rekursiv) vorgehen und in einer Schleife über die Spalten und die Punkte der Spalten die Kostenmatrix $c(i,j)$ füllen. Die jeweils gewählte Richtung wird in einer entsprechenden Matrix r gespeichert. Danach ist dann ausgehend vom optimalen Punkt der letzen Spalte eine Rückverfolgung des Pfades über die Matrix r möglich.

- Da die Pfade zyklisch sind, müssen wir in der obersten Zeile der Matrix als möglichen Vorgänger die Punkte der letzten Zeile berücksichtigen.
- Wenn die Transformation eine Spiegelung enthalten kann, kann sich die Punktreihenfolge invertieren. Wir müssen dann die Pfadsuche ein zweites Mal anwenden (von links unten nach rechts oben) und das Minimum der beiden Pfade wählen.
- Eigentlich müssten wir berücksichtigen, dass der Nachfolger des optimalen Punktes in der letzten Spalte zyklisch der Startpunkt in der ersten Spalte sein muss. Dies erhöht sofort die Komplexität des Algorithmus. Praktische Erfahrungen haben gezeigt, dass dies nicht nötig ist.

Die Komplexität des gesamten Algorithmus beträgt $O(m \cdot n)$.

22.1.3. Bestimmung der affinen Transformation

Durch den optimalen Pfad haben wir eine Liste von Punktreferenzen bestimmt, wobei es für einen Punkt durchaus mehrere Referenzpunkte geben kann. Da die mehrdeutigen Referenzpunkte Nachbarpunkte sein müssen, stören sie nicht und können in der Liste verbleiben. Nun können wir die affine Transformation bestimmen. Bei Verwendung der L^2-Norm führt dies auf die Gaußschen Normalengleichungen mit den 6 Unbekannten der affinen Transformation. Benutzen wir dagegen die L^1-Norm, dann ist dies numerisch aufwendiger und wir benutzen die lineare Programmierung dazu, siehe [44].

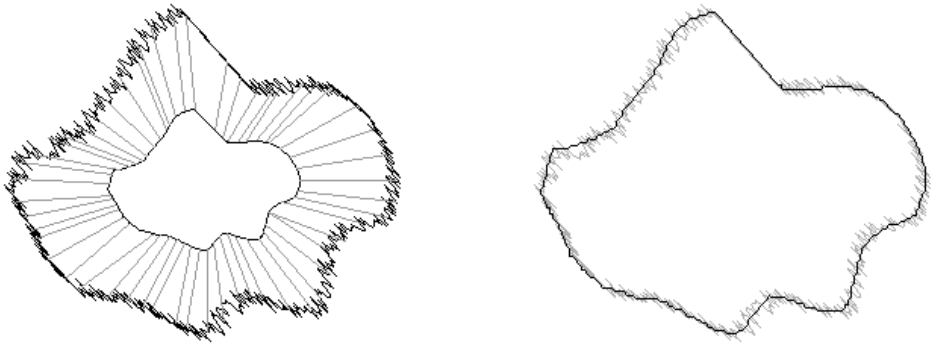


Abbildung 22.2.: Referenzpunkte und affin transformiertes Objekt

Im Beispiel in der Abbildung 22.2 wurde die innere Kontur affin transformiert und mit Rauschen überlagert. Man sieht die Referenzen und das Ergebnis des Matchings. In der Abbildung 22.3 wurde die innere Kontur transformiert und mit einer starken Störung versehen. Das Matchingresultat ist trotzdem nur wenig gestört, was die Robustheit dieses Verfahrens zeigt.

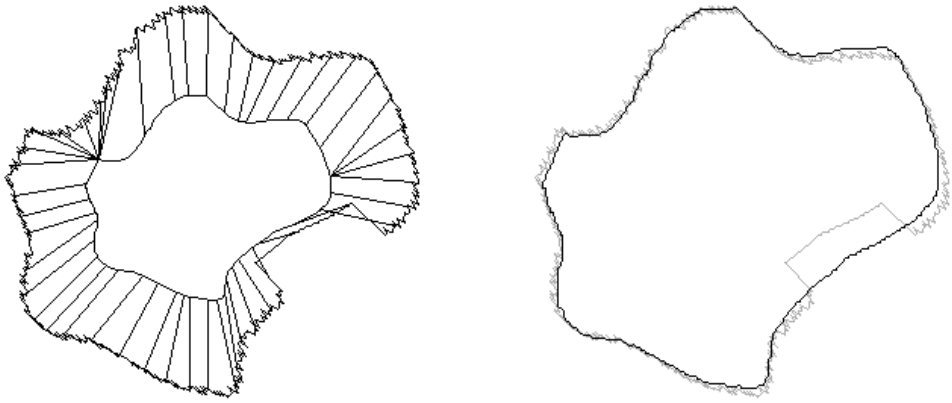


Abbildung 22.3.: Referenzpunkte und affin transformiertes Objekt

22.2. Ungeordnete Mengen von Referenzpunkten

Sind die Punkte nicht geordnet kann die dynamische Programmierung nicht mehr zur Zuordnung verwendet werden. Wir benutzen zum Beispiel den ICP-Algorithmus oder die Ungarische Methode.

Teil X.

Anhang

Literaturverzeichnis

[1] Ahmed N., Natarajan T., Rao K.R.: Discrete Cosine Transform. IEEE Trans. Computers (1974) 90-93.

[2] M.J. Atallah (Ed.), Algorithms and Theory of Computation. CRC Press, 1999

[3] B. P. Bogert, M. J. R. Healy, J. W. Tukey: The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking. Proceedings of the Symposium on Time Series Analysis, pp. 209-243, New York: Wiley, 1963.

[4] D. Boukerroui, J.A. Noble, M. Brady: On the Choice of Band-Pass Quadrature Filters. Journal of Mathematical Imaging and Vision, Vol. 21, No. 1 (2004) 53-80.

[5] Bose T.: Digital Signal and Image Processing. John Wiley and Sons 2004.

[6] Butz T.: Fouriertransformation für Fußgänger. Teubner 2003.

[7] W.T. Cathey and E.R. Dowski: New Paradigm for Imaging Systems. Appl. Optics 41 (2002) 6080-6092.

[8] S. Chaudhuri, A.N. Rajagopalan: Depth From Defocus: A Real Aperture Imaging Approach. Springer 1999.

[9] J. W. Cooley and J. W. Tukey: An algorithm for the machine calculation of complex Fourier series. Math. Comp. 19 (1965), 297-301

[10] Chen Q.G., Defrise M., Donconinck F.: Symmetric phase-only matched filtering of fourier-mellin transforms of image registration and recognition. IEEE PAMI 16 (1994) 1156-1168.

[11] D.E. Dudgeon, R.M. Mersereau: Multidimensional Digital Signal Processing. Prentice-Hall, 1984.

[12] Fenimore E.E, Cannon T.M.: Coded aperture imaging with uniformly redundant rays. Appl. Optics 17 (1978) 337.

[13] R.W. Frischholz, K.P. Spinnler: Class of algorithms for realtime subpixel registration. Proceedings Europto-Conference, München, Juni 1993, pp. 1-10.

[14] E. Rahtu, M. Salo, J. Heikkilä: Affine Invariant Pattern Recognition Using Multiscale Autoconvolution. PAMI Vol. 27, No. 6 (2005) 908-918.

[15] R.C. Gonzales, R.E. Woods: Digital Image Processi9ng. Third Edition, Pearson 2008.

[16] Granlund G.H., Knutsson H.: Signal Processing for Computer Vision. Kluwer Academic Publishers 1995.

[17] Multi-frame image restoration and registration. Advances in Computer Vision and Image Processing 1, (1984) 317-339.

[18] M.K. Hu: Visual pattern recognition by moment invariants. IT 8 (1962) 179-187.

[19] Jaroslawski: Einführung in die Digitale Bildverarbeitung. Deutscher Verlag der Wissenschaften, Berlin 1985.

[20] Bernd Jähne: Digitale Bildverarbeitung. Springer 2005

[21] Klette R. and A. Rosenfeld: Digital Geometry. Elsevier 2004.

[22] Kruger S., Calway A.: A multiresolution frequency domain method for estimating affine motion parameters. Proceedings ICIP 1996, Vol. I, pp. 113-116.

[23] Kuglin C.D., Hines D.C.: The phase correlation image aligment method. Proc. IEEE on Cybernetics and Society, New York 1975, pp. 163-165.

[24] Iterative Indentification and Restoration of Images. Kluwer Academic Publishers,1991.

[25] Lucy L.B.: An Iterative Technique for the Rectification of Observed Distributions. The Astronomical Journal 74 (1974) 745-754.

[26] L. Lucchese, G.M. Cortelazzo, C. Monti: Estimation of affine transformations between images pairs via fourier transform. Proceedings ICIP, Vol. III, 1996, pp. 715-718.

[27] J. Matas, O. Chum, M. Urban, T. Pajdla: Robust wide baseline stereo from maximally stable extremal regions. Proc. BMVC, 2002, pp. 384-393.

[28] R. Mukundan, K.R. Ramakrishnan: Fast Computation of Legendre and Zernike Moments. PR 28 (1995) 1433-1442.

[29] M.C. Morrone, D.C. Burr: Feature detection in human vision: A phase-dependent energy model. Proc. R. Coc. Lond. B, Vol. 235, (1988), 221-245.

[30] A. V. Oppenheim: Superposition in a class of nonlinear systems. Ph.D. diss., Res. Lab. Electronics, M.I.T. 1965

[31] A. V. Oppenheim, R. W. Schafer: Digital Signal Processing. Prentice Hall 1975.

[32] Super-Resolution Image Reconstruction: A Technical Overview. IEEE Signal Processing Magazine (2003) 21-36.

[33] S.K. Nayar, Y. Nakagawa: Shape from Focus. PAMI 16 (1994) 824-831.

[34] Richardson W.H.: Bayesian-based iterative Method of Image Restoration. Journal of the Optical Society of America, 62, (1972) 55-59.

[35] E. Rodner, H.Suesse, W. Ortmann, J. Denzler: Difference of Boxes Filters Revisited: Shadow Suppression and Efficient Character Segmentation. Proc. 8th IAPR Int. Workshop on Document Analysis Systems, Japan 2008, pp. 263-269.

[36] Rosenfeld A., Thurston M.: Edge and Curve Detection for Visual Scene Analysis. IEEE Trans. on Computers 20 (1971) 562-569.

[37] H. Suesse, W. Ortmann: Robust Matching of Affinely Transformed Objects. Proc. ICIP 2003, Part II, 383-386.

[38] Suesse H., Ditrich F.: Robust Determination of Rotation-Angles for Closed Regions using Moments. Proc. ICIP 2005, pp. 337-340.

[39] H. Süße, E. Rodner: Bildverarbeitung und Objekterkennung, Springer Vieweg, 2014

[40] Trummer M., Suesse H., Denzler J.: Coarse Registration of 3D Surface Triangulations Based on Moment Invariants with Applications to Object Alignment and Identification. Proceedings ICCV 2009, Japan, pp. 1273-1279.

[41] Umbaugh S.E.: Computer Imaging - Digital Image Analysis and Processing. CRC Press 2005.

[42] P. H. van Cittert, Zum Einfluß der Spaltbreite auf die Intensitätsverteilung in Spektrallinien. II. Z. Physik 69 (1931), pp. 298–308.

[43] P. Vanderwalle, S. Süsstrunk, and M. Vetterli: A Frequency Domain Approach to Registration of Aliased Images with Application to Super-resolution. EURASIP, Vol. 2006, Art. ID 71459,, pp. 1-14.

[44] K.Voss, H.Suesse: Affine Point Pattern Matching. Pattern Recognition. Lecture Notes in Computer Science 2191, 2001, pp. 155-162.

Index