Bildverarbeitung

Fragen

Manuel Pauli Sebastian Schweikl Thomas Lang

24. Juni 2016

1 Allgemein

Erkläre Operationen: Skalierung, Translation

Erkläre das Prinzip einer Lochkamera Eine Lochkamera ist der Vorläufer einer modernen Kamera. Dabei fällt das natürliche Licht durch ein kleines Loch (die Blende) ein. Das Bild trifft dann auf der Rückwand der Kamera auf. Bedingt durch die Geometrie steht das Bild auf dem Kopf, dessen Höhe/Breite wird dabei durch den Abstand des Objektes zur Blende, des Abstandes der Blende zur Rückwand und von der Größe des Objektes und der Blende bestimmt.

2 Fourier

2.1 Fouriertransformation

Nenne die Gleichung Für eine Funktion $f \in L_1(\mathbb{R})$ ist die Fouriertransformation durch

$$\hat{f}(\xi) = \int_{\mathbb{R}} f(t) e^{-i\xi t} \mathrm{dt}$$

für alle $\xi \in \mathbb{R}$ definiert.

Das ist die Gleichung für L_1-Funktionen. Wie sieht das im L_2 aus? Die Fouriertransformation im Signalraum L_2 kann ganz analog zum obigen Fall verwendet werden. Die resultierende Fouriertransformierte liegt dann in $L_1(\mathbb{R}) \cap L_2(\mathbb{R})$.
Formal beruht dies darauf, dass der Raum $L_1(\mathbb{R}) \cap L_2(\mathbb{R})$ dicht in $L_2(\mathbb{R})$ liegt. Deshalb lässt sich also immer eine Funktionenfolge aus $L_1(\mathbb{R}) \cap L_2(\mathbb{R})$ finden, die im Grenzwert gegen unsere gesuchte Fouriertransformierte konvergiert[1].

Wie sieht das Ergebnis der FT bei einer reellwertigen Funktion aus?

[1] Geht immer, zwecks Banachraum und so.

Was sagt der Satz von Parseval/Plancherel? Der Satz von Parseval/Plancherel beschreibt den Wechsel zwischen einem Skalarprodukt im Zeit- und einem Skalarprodukt im Frequenzbereich.
Formal gilt für Funktionen $f, g \in L_1(\mathbb{R})$:

$$\int_{\mathbb{R}} f(t)g(t)\mathrm{d}t = \frac{1}{2\pi}\int_{\mathbb{R}} \hat{f}(\vartheta)\overline{\hat{g}(\vartheta)}\mathrm{d}\vartheta$$

Im Speziellen für $f = g$ folgt, dass

$$\|f\|_1^2 = \frac{1}{2\pi}\|\hat{f}\|_2^2$$

gilt, also dass hier (bis auf Normierung) eine Isometrie vorliegt.

Sie sprachen von Isometrie, warum ist dann der konstante Faktor vor dem rechten Term? Dieser Faktor kommt daher, weil wir bei der Definition der Fouriertransformation keinen Faktor dabei hatten. Hätte man dort einen Faktor $1/\sqrt{2\pi}$ hinzugefügt, so hätte man hier eine perfekte Isometrie, was man ja mit Energieerhaltung[2] identifizieren kann.

Wie sieht die inverse Fouriertransformation aus? Unter der Voraussetzung, dass sowohl für f als auch für $\hat{f}$ gilt, dass diese Funktionen aus L_1 sind, ist die inverse Fouriertransformation definiert als

$$f(\xi) = \left(\hat{f}\right)^{\wedge}(\xi) = \frac{1}{2\pi}\int_{\mathbb{R}} \hat{f}(t)e^{i\xi t}\mathrm{d}t$$

Zusammenhang zwischen Faltung und Fouriertransformation Ein wichtiger Zusammenhang ist, dass sich die Faltung zweier Funktion als Produkt ihrer Fouriertransformierten darstellen lässt (vorausgesetzt deren Existenz).
Es gilt also:

$$\begin{aligned}
\widehat{(f*c)}(\xi) &\overset{\text{def}}{=} \int_{\mathbb{R}} (f*c)(t)e^{-i\xi t}\mathrm{d}t \\
&\overset{\text{def}}{=} \int_{\mathbb{R}}\int_{\mathbb{R}} f(t-x)c(x)\mathrm{d}x e^{-i\xi t}\mathrm{d}t \\
&\overset{\text{Tonelli}}{=} \int_{\mathbb{R}}\int_{\mathbb{R}} f(t-x)c(x)e^{-i\xi t}\mathrm{d}x\mathrm{d}t \\
&\overset{t=t+x}{=} \int_{\mathbb{R}}\int_{\mathbb{R}} f(t)c(x)e^{-i\xi(t+x)}\mathrm{d}x\mathrm{d}t \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(t)e^{-i\xi t}c(x)e^{-i\xi x}\mathrm{d}x\mathrm{d}t \\
&\overset{\text{Tonelli}}{=} \int_{\mathbb{R}} f(t)e^{-i\xi t}\mathrm{d}t \int_{\mathbb{R}} c(x)e^{-i\xi x}\mathrm{d}x \\
&= \hat{f}(\xi)\hat{c}(\xi)
\end{aligned}$$

[2] Die 2-Norm $\|\cdot\|_2$ beschreibt ja i.W. Energie.

Erkläre den Weg zur FFT Der Weg von der DFT zur FFT ist der, dass man einen Baumalgorithmus implementiert. Konkret arbeitet ein Teilbaum alle Folgenglieder mit geradem Index ab, während der andere Teilbaum den Rest bearbeitet, wobei sich natürlich die Indexmenge in den Teilbäumen vom Vaterknoten verändert.

Formal habe der Vaterknoten die Indexmenge $\mathbb{Z}_n = \{0, \ldots, n-1\}$ zur Folge c und sei $n = 2 * m, m \in \mathbb{N}$. Dann ergibt sich für die Teilbäume, dass der eine alle Folgenglieder $c(2k)$ und der andere alle Folgenglieder $c(2k+1), k \in \mathbb{Z}_m = \{0, \ldots, m-1\}$ verarbeitet.

Wie schnell ist die FFT? Da man dabei einen Baumalgorithmus implementiert, hat man also *nur* $\log n$ Ebenen abzuarbeiten, wobei man immer noch alle Folgenglieder $c(k), k \in \mathbb{Z}_n$ verarbeiten muss, es entsteht also ein Aufwand in $O(n \log n)$. Nachzuweisen ist dies mit dem Mastertheorem.

Theorie schön und gut, aber wozu braucht man die Fouriertransformation nun in der Praxis? Mit der ganzen Theorie (insbesondere mit dem Zusammenhang zwischen der Faltung und der Fouriertransformation) können Bildverarbeitungsalgorithmen in verschiedenster Hinsicht optimiert werden. Beispielsweise um die reine Berechnung zu beschleunigen (Multiplikation vs Faltung) oder um die numerische Stabilität zu erhöhen.

Der Standardweg ist dabei, dass man das Signal fouriertransformiert, auf dem Ergebnis die Berechnungen durchführt und anschließend wieder zurücktransformiert.

Wie funktioniert das JPEG-Kompressionsverfahren mit Hilfe der DFT? Die Idee ist hierbei, die Tatsache der Abhängigkeitsfreiheit zu nutzen um den Algorithmus massiv parallel zu machen. Dabei ist noch nicht mal die FFT gemeint, sondern das Vorgehen das Bild in Makroblöcke der fixen Größe $8x8$ aufzuspalten. Diese fixe Größe kommt daher, dass man also pro Makroblock 16 1×4- Vektoren vorliegen hat, welche z.B. auf Graphikkarten oder modernen CPUs sehr effizient verarbeitet werden können. Falls die Dimensionen des Bildes keine Vielfachen von 8 sind, so können die fehlenden Ränder z.B. auf 0 gesetzt oder das Bild periodisch fortgesetzt werden.

Hat man nun die Makroblöcke vorliegen, so wendet man auf diese die Transformation an (sei es die DFT, die FFT oder die DCT). Das Ergebnis wird dann mit einem Kompressionsverfahren verknüpft, um die Größe jedes Makroblockes (und damit die Bildgröße) dramatisch zu verringern. Beispiele dafür wäre z.B. der Huffman-Code, welcher eindeutige Codes unterschiedlicher Länge generiert, die die ursprünglichen Werte darstellen. Dieses Encoding wäre sogar verlustfrei.

Bei der Dekompression geht man entsprechend umgekehrt vor: Man Ersetzt die gekürzten Codes mit den ursprünglichen Werten und wendet die inverse Transformation (IDFT, IFFT, IDCT) an, um das alte Bild z.B. darstellen zu können.

Macht das JPEG wirklich so? Nein, da die DFT/FFT u.U. imaginäre Werte liefert. Diese lassen sich aber nur auf wenigen (heute nicht mehr aktuellen) Maschinen in Hardware darstellen, und die Verarbeitung in Software dauert sehr lange.

Als Alternative wird daher in JPEG (vor Standard JPEG-2000) die *Diskrete Cosinus-Transformation*(DCT) verwendet, welche den selben Effekt wie die DFT erzielt, aber nur reelle Werte liefert.

Wo liegen die Informationsmaxima? Die Informationsmaxima liegen am Rand, da die höherfrequenten Anteile dort liegen.

Nenne die Formel der DFT

Fouriertransformation im $\mathbb{R}^2$ Die Fouriertransformation im zweidimensionalen Raum ist entsprechend die Variante für Bilder.
Hierbei hat man also Punkte $p = (x, y) \in \mathbb{Z}^2$, entsprechen betrachtet man auch die Fouriertransformation in zwei unterschiedlichen Richtungen (in x- und in y-Richtung).
Formal übersetzt ergibt sich also die Fouriertransformation im $\mathbb{R}^2$ als

$$\hat{f}(\xi) = \int_{\mathbb{R}}^{2} f(t) e^{-i\xi^T t} \mathrm{dt} \quad , \xi, t \in \mathbb{R}^2$$

2.2 Faltung

Was ist formal eine Faltung? Formal gesehen ist eine Faltung ein Integral. Dieses ist für Funktionen $f, g \in \mathbb{R}^n \to \mathbb{C}$ definiert als

$$(f * g) = \int_{\mathbb{R}^n} f(\cdot - t) g(t) \mathrm{dt}.$$

Prinzipiell ist dieser Ausdruck definiert für alle Funktionen, die für *fast alle* $x \in \mathbb{R}$ wohldefiniert sind. In unserem eingeschränkten Kontext der integrierbaren Funktionen ist dies automatisch erfüllt.

Wie kann man sich so eine Faltung graphisch vorstellen? Man kann sich das so vorstellen, dass man eine Funktion über die zweite *schiebt* und dort die Übereinstimmung berechnet. Bei einer hohen Übereinstimmung zwischen den Funktionen wird dieses Integral einen hohen Wert liefern, während sich bei wenig Übereinstimmung nur ein kleiner Wert ergibt.

Was bringt so eine Faltung? Das im letzten Punkt beschriebene Verhalten der Faltung kann man sich z.B. in der Texterkennung zu nutze machen. Dabei prüft man z.B., ob es eine hohe Übereinstimmung zwischen einem Bild eines Buchstaben und dem vorliegenden Bild gibt, indem man über diese Bildteile die Faltung berechnet. Ein hoher Wert indiziert dabei, dass mit hoher Wahrscheinlichkeit der gesuchte Buchstabe vorliegt.
Ganz allgemein ist dieses Prinzip die Grundlage von Filterungen.

3 Filter

Was ist ein Filter? Ein Filter ist ein Operator, der ein gegebenes Signal in ein anderes Signal transformiert.

Wie sieht ein Filter allgemein aus?

Wie sieht ein Tiefpass-Filter aus? Zeichne den Graph

Forderungen an die Transferfunktion für Tiefpass

Erkläre das Gibbs-Phänomen

Welche Arten von Filter gibt es? Erklären Sie diese

Zeichen Sie ein Schaltbild zu einem Filter (Addierer,Verzögerer, Multiplizierer)

Welchen Filter zur Kantenerkennung?

Was ist eine Impulsantwort?

Spielt die Laufzeit von Filtern in der Praxis eine Rolle?

3.1 Was ist ein Gradientenfilter?

Wozu ist er gut? Mit einem Gradientenfilter können Kanten in Bildern erkannt/hervorgehoben werden.

Wie funktioniert er? Ist eine solche Kante in einem Bild, so muss dort ein signifikant anderer Farbwert vorhanden sein. Dies macht man sich zu nutze, indem man sich die Steigung zwischen zwei benachbarten Farbwerten ansieht: Ist diese Steigung groß, so sind die Farbwerte grob unterschiedlich (beispielsweise eine schwarze Kante auf einem weißen Hintergrund), was ein Indikator für eine Kante ist.

Wie sieht so ein Filter aus? (Gradient + Filtermatrix) Wie schon erwähnt betrachtet man die Steigungen. Kontinuierlich betrachtet ist dies aber gerade die erste Ableitung, was im mehrdimensionalen der Gradient

$$\nabla f = \begin{pmatrix} \frac{\partial}{\partial x_1} f \\ \vdots \\ \frac{\partial}{\partial x_n} f \end{pmatrix}$$

ist. Um diesen nun auf ein diskretes Signal (z.B. ein Bild) anwenden zu können, muss man diesen geeignet diskretisieren. Dies ergibt die Filtermatrizen in x- und y-Richtung

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Wie kommt man auf die Impulsantwort?

Nachteile + mögliche Gegenmaßnahmen Ein wesentlicher Nachteil eines reinen Gradientenfilters ist, dass dieser gegen Rauschen extrem empfindlich ist, da auch diese feinen Unterschiede unterschiedliche Farbwerte und damit potenzielle Kanten liefern.
Um dies zu beheben, kann man den Gradientenfilter mit einem Mittelwertfilter kombinieren, welcher das Bild glättet und damit Rauschen entfernt, gefolgt vom eigentlichen Gradientenfilter.

Wie sieht so ein Filter aus? Obiges Vorgehen liefert z.B. diese neuen Filtermatrizen:

$$\frac{1}{9}\begin{pmatrix} -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

So viel zu Gradientenfilter, was ist aber dann ein Laplace-Filter? Ein Laplace-Filter ist eine Diskretisierung des Laplace-Operators

$$\Delta = \langle \nabla, \nabla \rangle,$$

welcher offensichtlich beide Richtungen kombiniert. Eine Diskretisierung ergibt z.B. die Filtermatrix

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

3.2 Filterbänke

Skizziere eine Filterbank und beschreibe den Vorgang

Nenne 3 typische Filter

Was ist die zentrale Eigenschaft von Filterbänken? Welche Voraussetzung muss dazu gelten?

Kann man Filterbänke hierarchisch aufbauen?

4 Abtastsatz

Welche Eigenschaften müssen für eine Abtastung gelten?

Erkläre den Shannonschen Abtastsatz

Wie funktioniert Abtasten überhaupt?

Was ist die kritische Abtastrate?

Wie berechnet man die kritische Abtastrate?

Was ist ein bandbeschränktes Signal?

Was ist der Träger eines Signals?

5 Hough-Transformation

Was ist die Hough-Transformation? Die Hough-Transformation ist ein allgemeines Verfahren, um in einem **binearisierten** Bild Formen zu erkennen.

Wie funktioniert sie? Bei der Hough-Transformation wird für alle möglichen Variationen einer Form im gesamten Bild gesucht, ob diese vorkommt. Dies wir dadurch realisiert, dass man bei der Test-Form alle Pixel gezählt werden, bei der in der binearisierten Form ein Farbwert (z.B. 1 für Schwarz) steht. Hat man nun eine hohe Anzahl gefunden, so hat man ziemlich sicher auch die gesuchte Form im Bild erkannt.

Was ermöglicht sie? Mit diesem Verfahren kann man nun nach *allen möglichen* Formen in Bildern suchen, die Formen müssen sich entsprechen parameterisieren lassen. Dies ist u.a. für Linien, Kreise, Ellipsen u.v.m. möglich.

Man beachte aber, dass man z.B. bei Linien zwei Freiheitsgrade hat. D.h. man muss für alle Möglichkeiten dieser Freiheitsgrade das Bild absuchen, was hier schon quadratischen Aufwand liefert. Will man nun z.B. Kreise finden, so hat man schon drei Freiheitsgrade (die x- und die y-Koordinate des Mittelpunktes und den Radius r), was entsprechend kubischen Aufwand liefert.

Was sind mögliche Optimierungsmöglichkeiten? Mögliche einfache Optimierungen sind z.B., dass man die Freiheitsgrade einschränkt. Sucht man z.B. in einem Bild nur nach Kreisen mit einem festen Radius, so besteht die einzige Variationsmöglichkeit in den Koordinaten des Kreismittelpunktes, was insgesamt also quadratischen Aufwand verursacht.

Welcher Filter spielt dabei eine Rolle?

Gibt es weitere Filter zur Kantenerkennung?

6 Muss noch zugeordnet werden

Erklären Sie die Heisenbergboxen