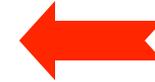


2. Digitale Codierung und Übertragung

2.1 Informationstheoretische Grundlagen



2.1.1 Abtasttheorem

2.1.2 Stochastische Nachrichtenquelle, Entropie, Redundanz

2.2 Verlustfreie universelle Kompression

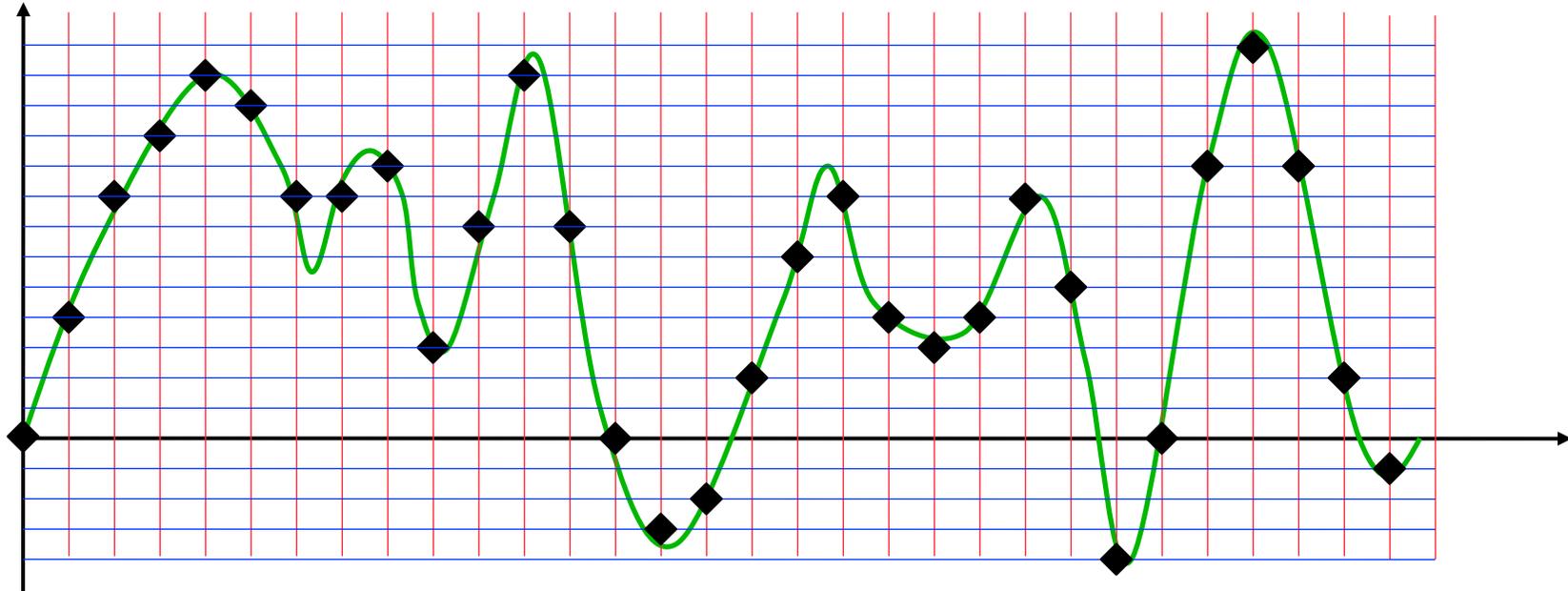
Siehe auch Medieninformatik-Buch Kapitel 2

Weiterführende Literatur zum Thema Informationstheorie:

Taschenbuch Medieninformatik Kapitel 2

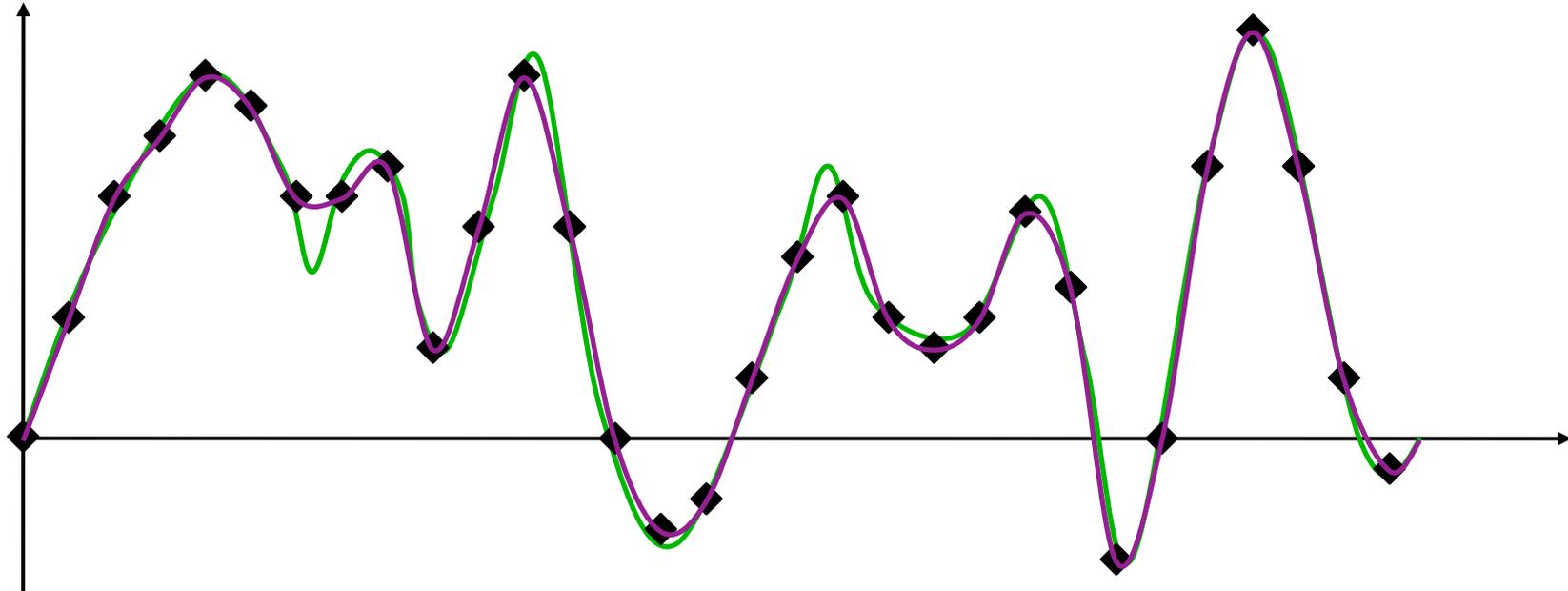
Herbert Klimant, Rudi Piotraschke, Dagmar Schönfeld:
Informations- und Kodierungstheorie, Teubner 2003

Digitalisierungsfehler (Wiederholung)



- Durch zu grobe Raster bei Diskretisierung und Quantisierung entstehen *Digitalisierungsfehler*.

Digitalisierungsfehler



- Fehlerklassen:
 - Zu grobe Quantisierung: Schlechtere Darstellung von Abstufungen
 - Zu grobe Diskretisierung, d.h. Fehler in der Abtastrate:
Zusammenhang schwerer zu verstehen; führt zu gravierenden Fehlern!

Abtastrate: Einführendes Beispiel

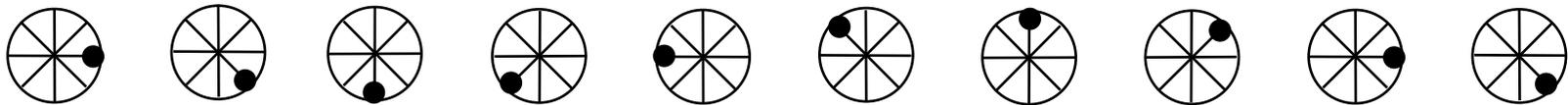
Warum drehen sich in Kinofilmen die Räder von Kutschen oft scheinbar rückwärts? <http://www.youtube.com/watch?v=0jL5qxx-cWI>



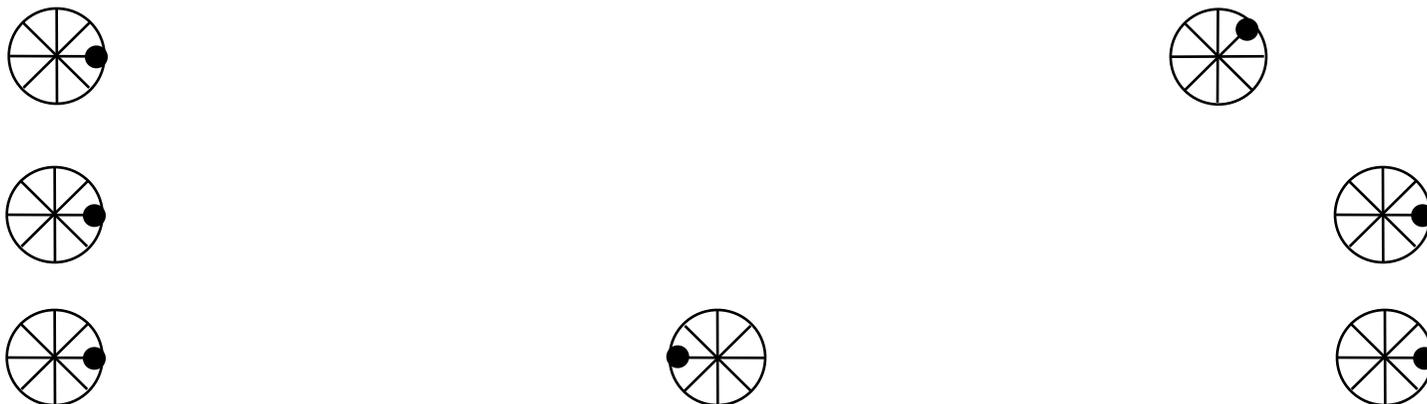
Abtastrate: Einführendes Beispiel

Warum drehen sich in Kinofilmen die Räder von Kutschen oft scheinbar rückwärts?

Rad (über die Zeit):



Aufnahmen (über die Zeit):



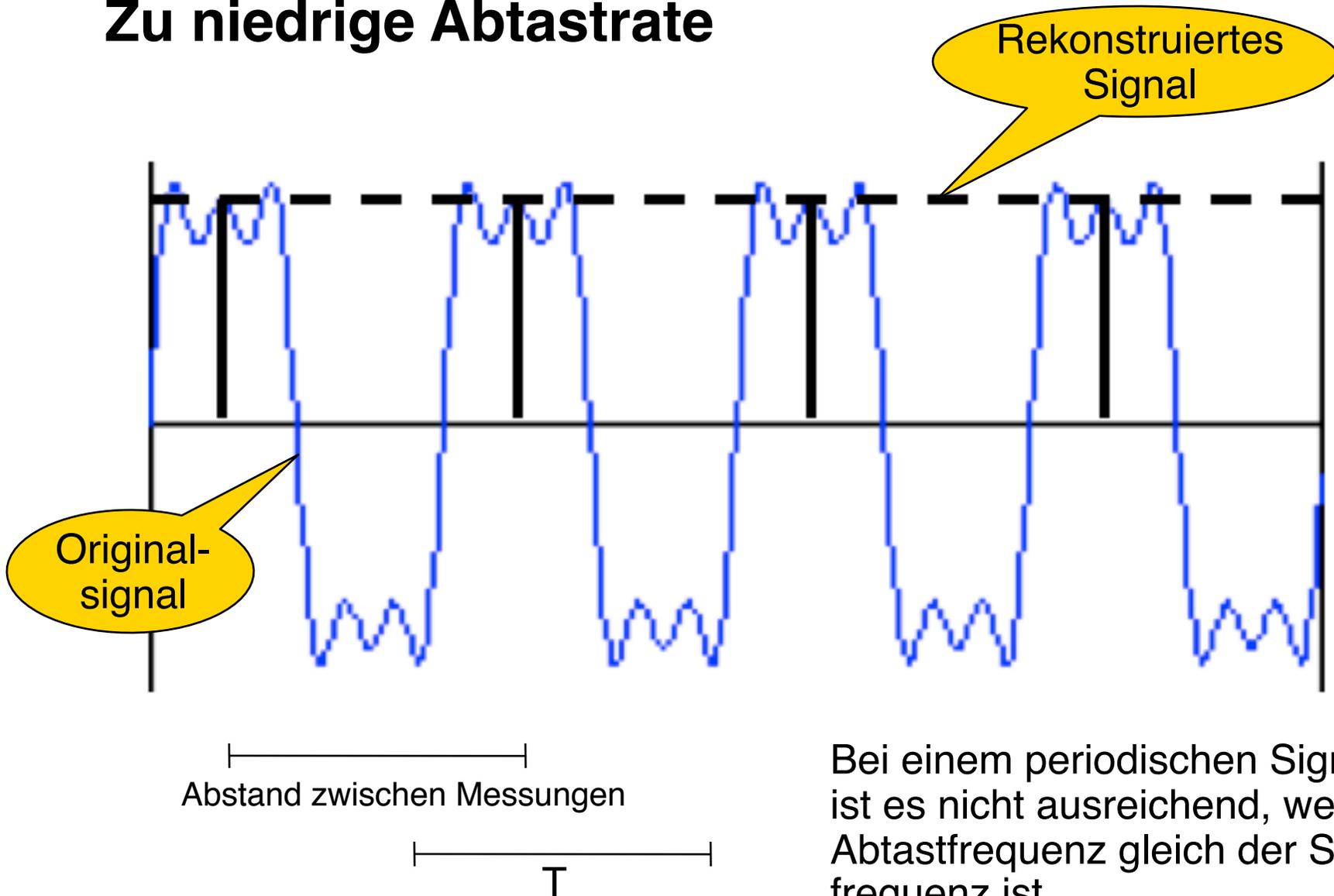
Frequenz

- Die Frequenz ist ein Maß für die Häufigkeit eines wiederkehrenden Ereignisses
- Maßeinheit:
 - *Hertz*, $1 \text{ Hz} = 1/\text{s}$
 - 1 Hz bedeutet einmal pro Sekunde
- Wiederkehr
 - Länge des Signalverlaufs bis zum Beginn der nächsten Wiederholung
 - Wellenlänge bei einer Sinusfunktion
 - Wiederkehr T bei gegebener Frequenz f :

$$T = \frac{1}{f}$$

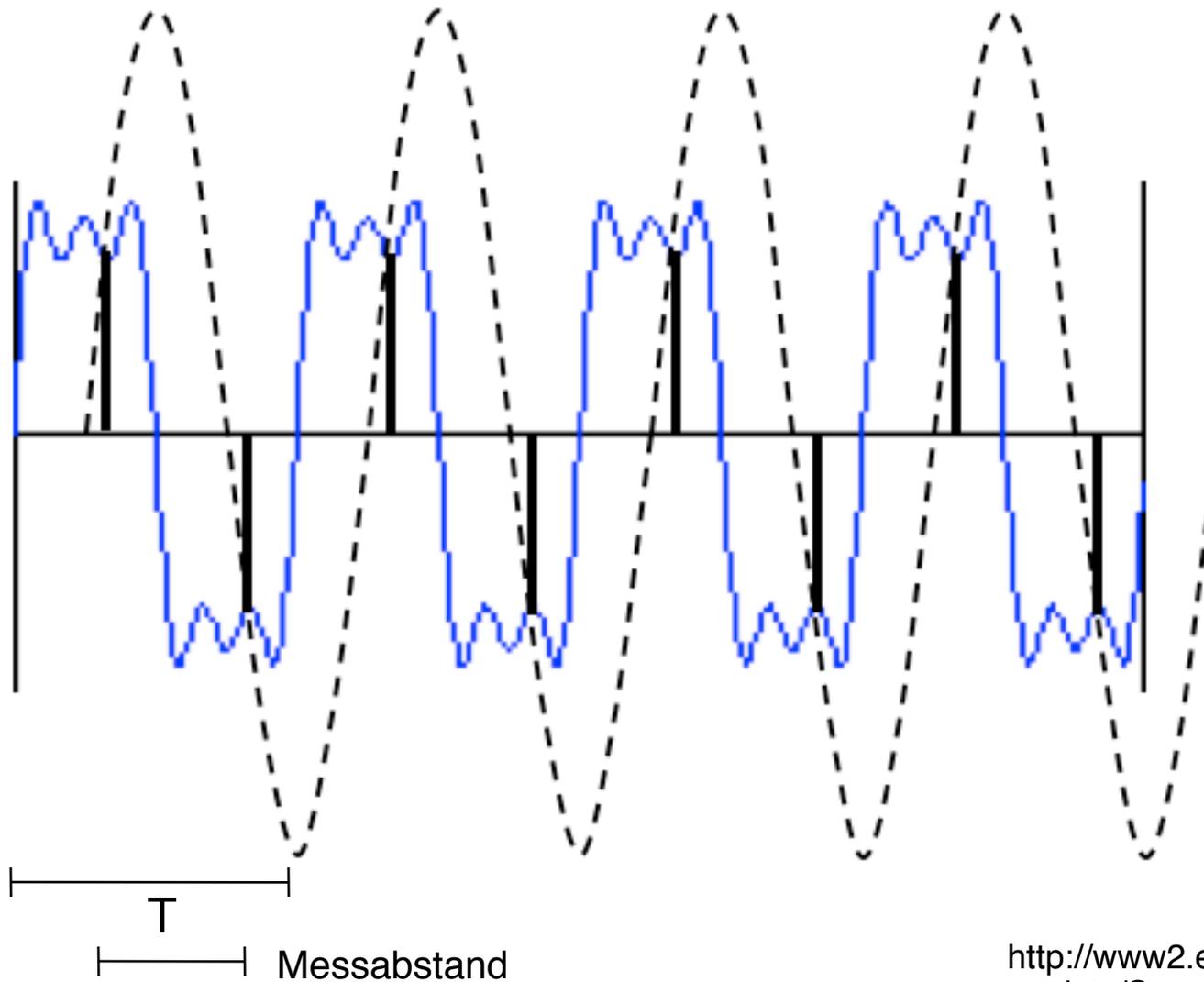
Hier zeitabhängige Signale – aber übertragbar auf raumabhängige Signale

Zu niedrige Abtastrate



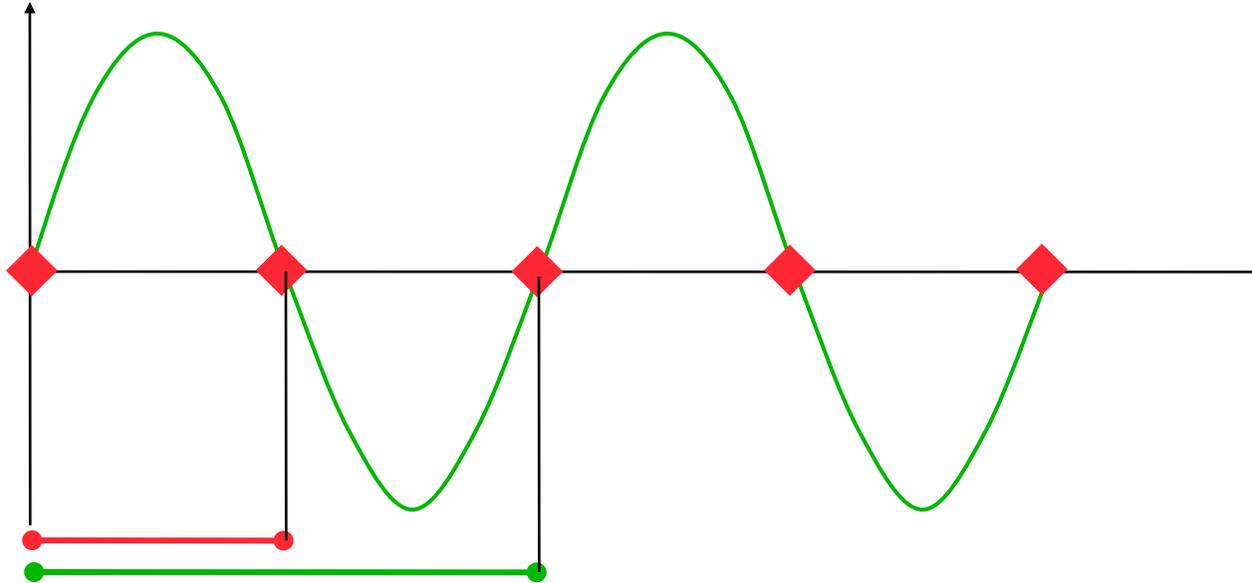
Bei einem periodischen Signal ist es nicht ausreichend, wenn die Abtastfrequenz gleich der Signalfrequenz ist.

Immer noch zu niedrige Abtastrate



<http://www2.egr.uh.edu/~glover/applets/Sampling/Sampling.html>

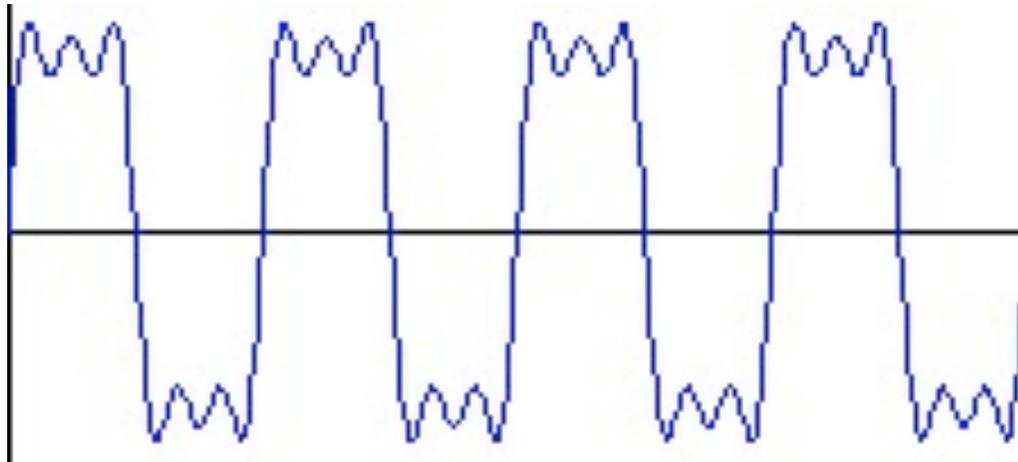
Wie groß muss die Abtastrate sein?



- Bei der doppelten Abtastrate gegenüber einer Sinus-Signalfrequenz ist die Abtastung „noch“ nicht korrekt.
- Mindestabtastung: Mehr als doppelte Frequenz im Vergleich zur Frequenz eines reinen Sinus-Signals

Bandbegrenzung

- Reale Signale bestehen immer aus einer Überlagerung von Signalanteilen verschiedener Frequenzen
- „Bandbreite“ = Bereich der niedrigsten und höchsten vorkommenden Frequenzen
 - Untere Grenzfrequenz
 - Obere Grenzfrequenz
- Grundfrequenz = Frequenz der Wiederholung des Gesamtsignals (bei periodischen Signalen)



Beispiel:

Überlagerung von Signalen mit 50 Hz (Grundfrequenz), 100 Hz und 150 Hz

Abtasttheorem

Nach Harry Nyquist (1928) oft auch Nyquist-Theorem genannt.
(Beweis von Claude Shannon)

Wenn eine Funktion

mit höchster vorkommender Frequenz f_g (Bandbegrenzung)

mit einer Abtastrate f_S abgetastet wird, so dass

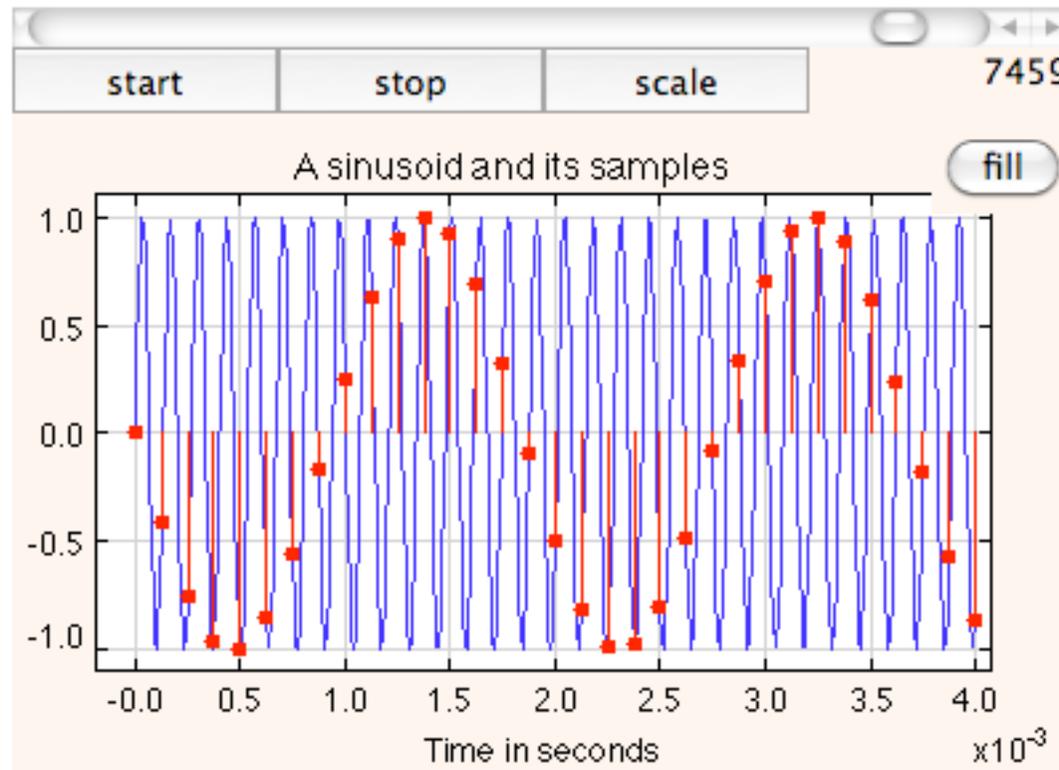
$$f_S > 2 \cdot f_g ,$$

dann kann die Funktion eindeutig aus den Abtastwerten
rekonstruiert werden.

Praktisches Beispiel: Abtastrate für Audio-CDs ist 44,1 kHz
(eindeutige Rekonstruktion von Signalen bis ca. 22 kHz)

Aliasing: Audio-Beispiel

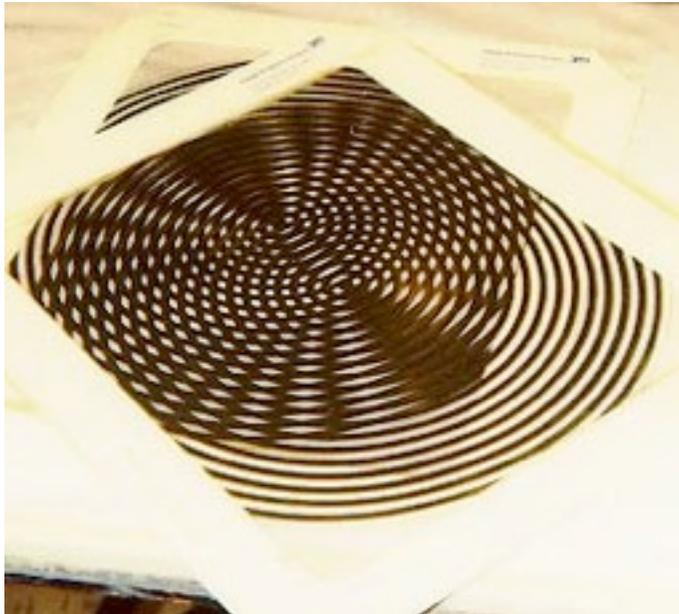
- Bei einer nicht genügend hohen Abtastrate entstehen Fehlinterpretationen der hochfrequenten Signalanteile (*Aliasing*)
- Beispiel Audio: Hohe Töne werden als tiefe Töne rekonstruiert.



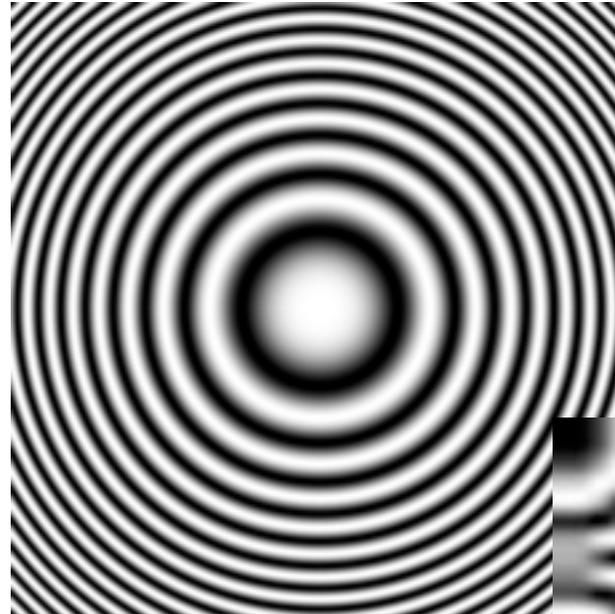
http://www.aw-bc.com/lee_varaiya/protected/week13/aliasing.html

Aliasing: Bildbeispiele

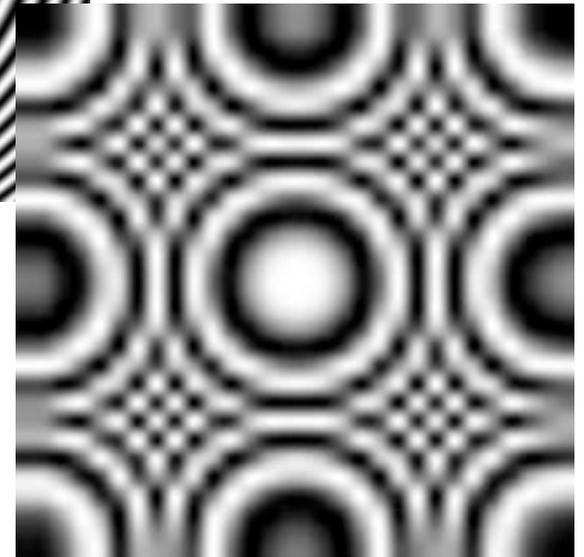
Bei Bildern liefert unzureichende Abtastung sogenannte *Moiré-Effekte*.



Moiré-Muster durch Überlagerung von Ringmustern
(physics.uiuc.edu)

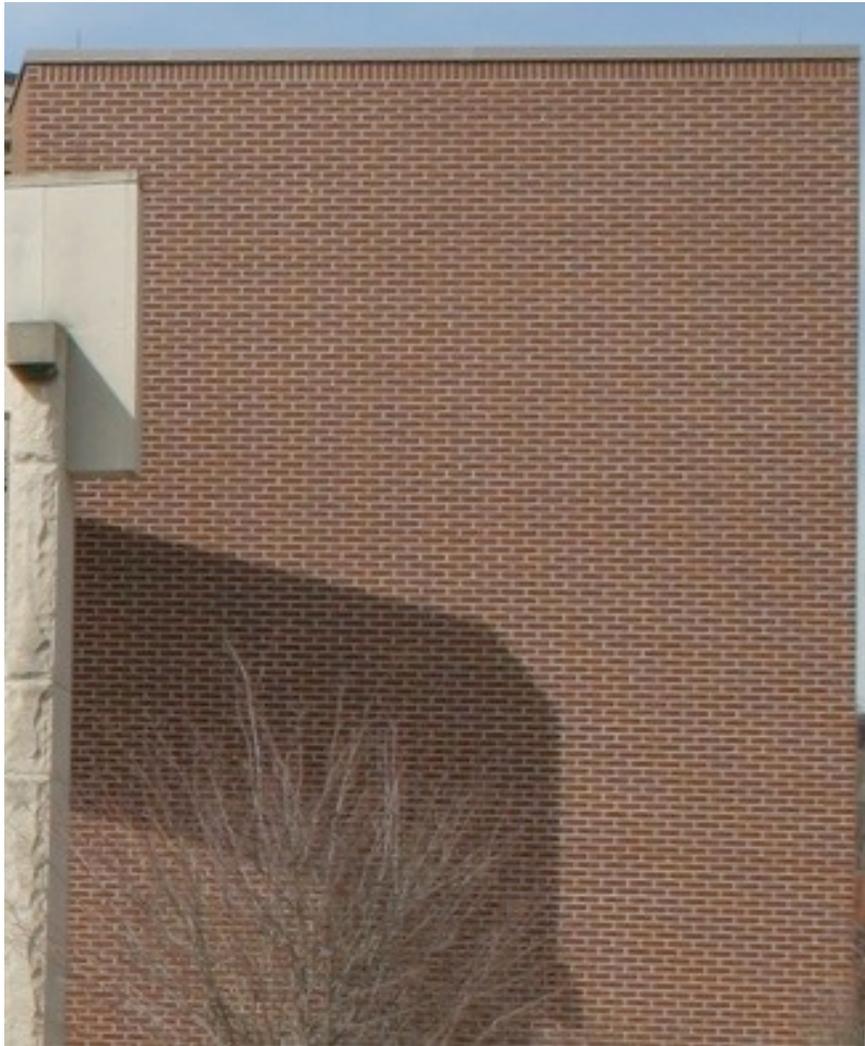


"Fresnel-Zonenplatte"



abgetastet mit 30
Punkten je Kante
und rekonstruiert

Moiré im Foto



Quelle:Wikipedia

Vermeidung von Aliasing: Filterung

- Vor digitaler Abtastung: Nyquist-Bedingung sicherstellen!
- Wenn höherfrequente Anteile ($\geq 1/2 f_S$) vorhanden,
 - Entfernen!
- Filterung
 - Bei Bildern und Ton anwendbar
- Anwendungsbeispiele:
 - Hochauflösendes Bild soll neu abgetastet werden
 - Signal aus einem Tongenerator soll abgetastet werden (z.B. Sägezahnsignal)

Wie perfekt ist die Rekonstruktion?

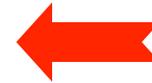
- Das Nyquist-Theorem ist ein mathematisches Theorem.
 - ***Keinerlei Verlust*** bei Rekonstruktion innerhalb der angegebenen Rahmenbedingungen
- Mathematische Rekonstruktion mit „idealem Tiefpass“
 - Siehe später!
- Praktische Rekonstruktion
 - Zum Teil sehr aufwändige Systeme für optimale Anpassung an Wahrnehmungsphysiologie
- Praktisches Beispiel:
 - Vergleich der Klangqualität von CD-Spielern (an der gleichen Stereoanlage)

2. Digitale Codierung und Übertragung

2.1 Informationstheoretische Grundlagen

2.1.1 Abtasttheorem

2.1.2 Stochastische Nachrichtenquelle, Entropie, Redundanz



2.2 Verlustfreie universelle Kompression

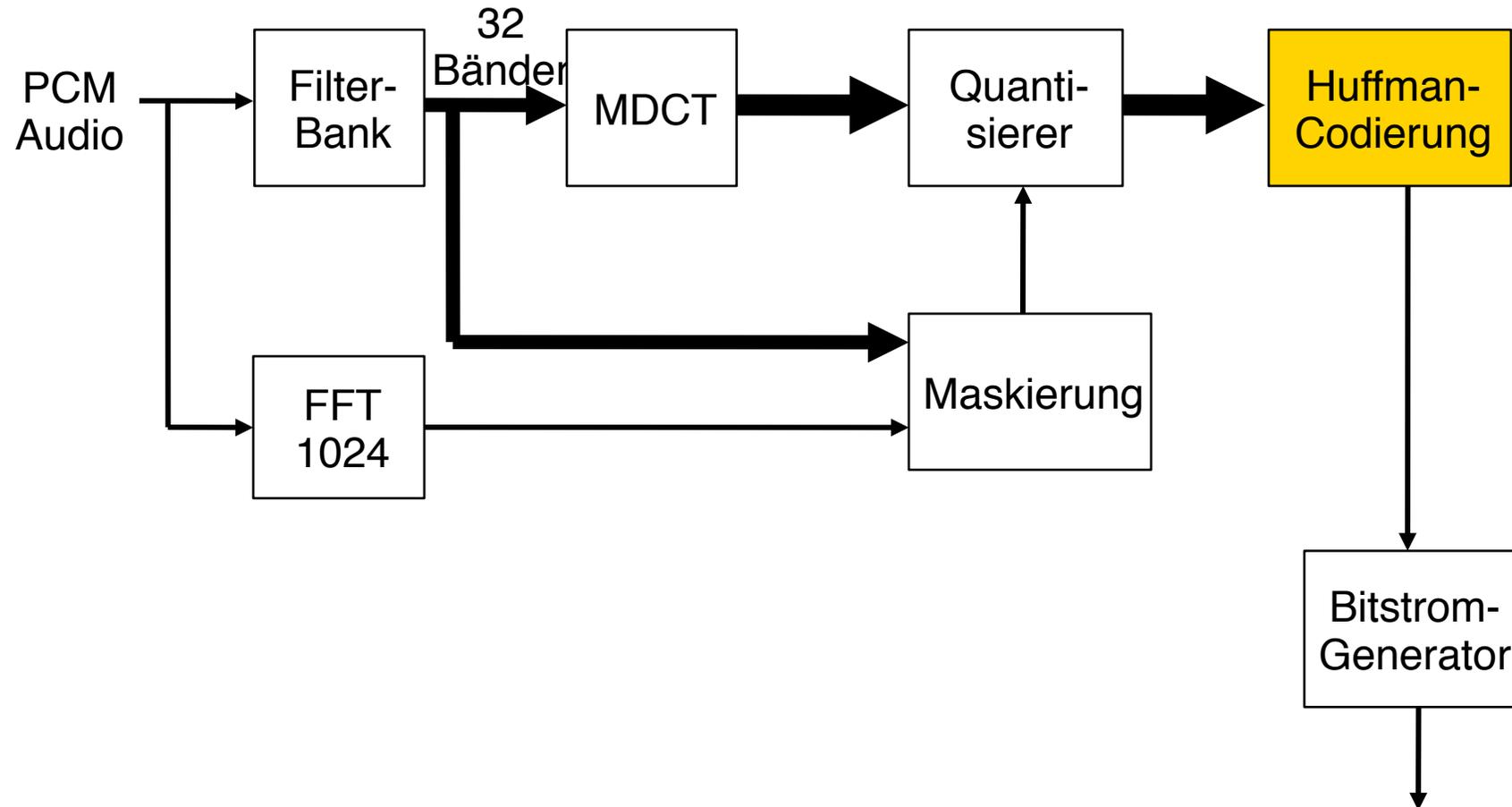
Weiterführende Literatur zum Thema Informationstheorie:

Taschenbuch Medieninformatik Kapitel 2

Herbert Klimant, Rudi Piotraschke, Dagmar Schönfeld:
Informations- und Kodierungstheorie, Teubner 2003

Einschub: Motivation für Informationstheorie

- Aufbau eines MPEG-Layer III (MP3) Encoders
 - Details siehe später!

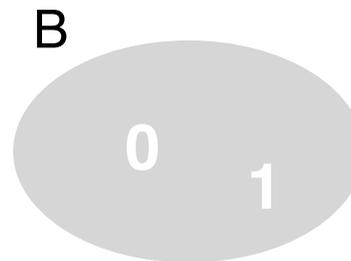
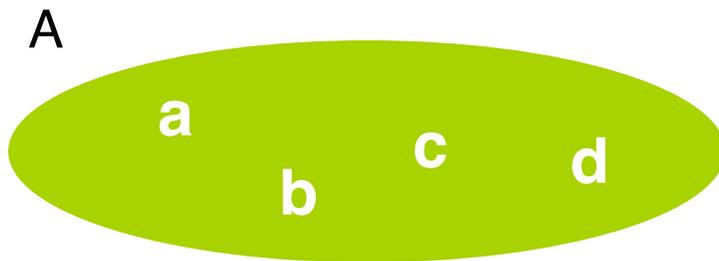


Stochastische Informationstheorie: Zeichenvorrat und Codierung

- Ein *Zeichenvorrat* ist eine endliche Menge von *Zeichen*.
- Eine Nachricht (im Zeichenvorrat A) ist eine Sequenz von Zeichen aus A
- Seien A und B Zeichenvorräte.
Eine *Codierung* c ist eine Abbildung von Nachrichten in A auf Nachrichten in B.

$$c: A \rightarrow B^* \quad (B^* : \text{Zeichenreihen über } B)$$

- Wir beschränken uns meist auf *binäre* Codierungen, d.h. $B = \{ 0, 1 \}$
- *Informationstheorie* (nach *Shannon*) betrachtet die Häufigkeit des Auftretens bestimmter Zeichen(folgen) in den Nachrichten einer Nachrichtenquelle.



Beispiel:

abca \rightarrow 00011000

ddc \rightarrow 111110

Entropie (1)

- Annahme *Stochastische Nachrichtenquelle*: Wir kennen die Häufigkeitsverteilung der Zeichen in den Nachrichten.
- *Entscheidungsgehalt (Entropie)* der Nachrichtenquelle:
 - Wie viele Ja/Nein-Entscheidungen (x_a) entsprechen dem Auftreten eines Einzelzeichens (a)?
 - Eine Ja/Nein-Entscheidung = 1 „bit“

- Beispiele:

Quelle 1	Zeichen a	A	B	C	D
	Häufigk. p_a	1	0	0	0
	x_a	0	-	-	-

Quelle 2	Zeichen a	A	B	C	D
	Häufigk. p_a	0.25	0.25	0.25	0.25
	x_a	2	2	2	2

p_a = Häufigkeit

x_a = Zahl der Entscheidungen

$$2^{x_a} = 1/p_a$$

$$x_a = \text{ld}(1/p_a)$$

(ld = Logarithmus zur Basis 2)

Entropie (2)

Durchschnittlicher Entscheidungsgehalt je Zeichen: Entropie H

$$H = \sum_{a \in A} p_a \text{ld} \left(\frac{1}{p_a} \right)$$

mit $x_a = \text{ld} (1/p_a)$: $H = \sum_{a \in A} p_a x_a$

Quelle 1	Zeichen a	A	B	C	D	$H = 0$
	Häufigk. p_a	1	0	0	0	
	x_a	0	-	-	-	

Quelle 2	Zeichen a	A	B	C	D	$H = 2$
	Häufigk. p_a	0.25	0.25	0.25	0.25	
	x_a	2	2	2	2	

Quelle 3	Zeichen a	A	B	C	D	$H = 1.75$
	Häufigk. p_a	0.5	0.25	0.125	0.125	
	x_a	1	2	3	3	

Entropie ist Maß für „Unordnung“, „Zufälligkeit“

Wortlängen und Redundanz

- Eine (Binär-)Codierung der Nachrichten einer stochastischen Nachrichtenquelle ergibt eine *durchschnittliche Wortlänge* L .

$$L = \sum_{a \in A} p_a |c(a)|$$

Quelle 2	Zeichen a	A	B	C	D	$H = 2$ $L = 2$
	Häufigk. p_a	0.25	0.25	0.25	0.25	
	Code $c(a)$	00	01	10	11	
Quelle 3	Zeichen a	A	B	C	D	$H = 1.75$ $L = 2$
	Häufigk. p_a	0.5	0.25	0.125	0.125	
	Code $c(a)$	00	01	10	11	

- **Redundanz = $L - H$**
- Redundanz ist ein Maß für die Güte der Codierung: möglichst klein!

Optimale Codierung

- Eine Codierung ist *optimal*, wenn die Redundanz 0 ist.
- Durch geeignete Codierung (z.B. Wortcodierung statt Einzelzeichencodierung) kann man die Redundanz beliebig niedrig wählen.
- Redundanz ermöglicht andererseits die Rekonstruktion fehlender Nachrichtenteile!
 - Beispiel: Natürlich Sprach
 - Beispiel: Fehlererkennende und -korrigierende Codes (z.B. Paritätsbits)

Quelle 3	Zeichen a	A	B	C	D	$H = 1.75$ $L = 2$
	Häufigk. p_a	0.5	0.25	0.125	0.125	
	Code $c(a)$	00	01	10	11	

Quelle 3	Zeichen a	A	B	C	D	$H = 1.75$ $L = 1.75$
	Häufigk. p_a	0.5	0.25	0.125	0.125	
	Code $c'(a)$	0	10	110	111	

2. Digitale Codierung und Übertragung

2.1 Informationstheoretische Grundlagen

2.2 Verlustfreie universelle Kompression



Weiterführende Literatur zum Thema Kompression:

Taschenbuch Medieninformatik Kapitel 2

Herbert Klimant, Rudi Piotraschke, Dagmar Schönfeld:
Informations- und Kodierungstheorie, Teubner 2003

Khalid Sayood: Introduction to Data Compression, 2nd. ed.,
Morgan Kaufmann 2000

Kompressionsverfahren: Übersicht

- Klassifikationen:
 - Universell vs. speziell
 - » Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
 - Verlustfrei vs. Verlustbehaftet
 - In diesem Kapitel: nur universelle & verlustfreie Verfahren
- Im folgenden vorgestellte Verfahren:
 - Statistische Verfahren:
 - » Huffman-Codierung 
 - » Arithmetische Codierung
 - Zeichenorientierte Verfahren:
 - » Lauflängencodierung (RLE Run Length Encoding)
 - » LZW-Codierung

Grundidee zur Huffman-Codierung

- Zeichen größerer Häufigkeit werden durch kürzere Codes repräsentiert
 - vgl. Morse-Code
- Das führt zu einem *Code variabler Wortlänge*:
 - Kein Codewort darf Anfang eines anderen sein (*Fano-Bedingung*)
- In optimalem Code müssen die beiden Symbole der niedrigsten Häufigkeit mit gleicher Länge codiert sein.

"Beweis"-Skizze:

- Wären die Längen verschieden, könnte man das längere Wort bei der Länge des kürzeren abschneiden
 - » Dann sind die beiden entstehenden Codes verschieden (sonst wäre Fano-Bedingung vorher verletzt gewesen)
 - » Kein anderes Codewort kann länger sein (da Zeichen niedrigster Häufigkeit), also kann die Kürzung nicht die Fano-Bedingung verletzen
- Dann hätten wir einen neuen Code mit kleinerer durchschnittlicher Wortlänge!

Huffman-Codierung (1)

- Gegeben: Zeichenvorrat und Häufigkeitsverteilung
- Ergebnis: Codierung
(optimal, wenn alle Häufigkeiten Kehrwerte von Zweierpotenzen sind)
- Wiederholte Anwendung dieses Schritts auf die Häufigkeitstabelle:
 - Ersetze die beiden Einträge niedrigster Häufigkeit durch einen Codebaum mit zwei Ästen „0“ und „1“ und trage die Summe der Häufigkeiten als Häufigkeit dafür ein.

Zeichen	A	B	C	D
Häufigkeit	0.5	0.25	0.125	0.125

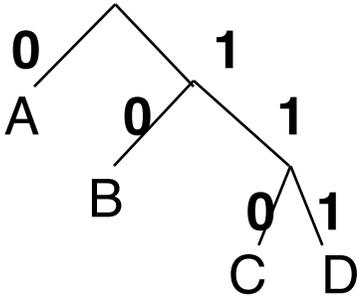
Zeichen	A	B		
Häufigkeit	0.5	0.25	0.25	

David Huffman 1951

Huffman-Codierung (2)

Zeichen	A	B	$\begin{array}{l} 0 \swarrow 1 \\ C \quad D \end{array}$
Häufigkeit	0.5	0.25	0.25

Zeichen	A	$\begin{array}{l} 0 \swarrow 1 \\ B \quad \begin{array}{l} 0 \swarrow 1 \\ C \quad D \end{array} \end{array}$
Häufigkeit	0.5	0.5



Resultierender Codebaum

Huffman-Codierung (3)

- Eine Nachricht, die sich an die gegebene Häufigkeitsverteilung hält:
ababacadaabacdba (Länge = 16 Zeichen)
- Codierung mit festen Wortlängen
(z.B. a = 00, b = 01, c = 10, d = 11)
Länge 32 bit
- Huffman-Codierung
(a = 0, b = 10, c = 110, d = 111)
0100100110011100100110111100
Länge 28 bit (d.h. ca. 12.5% Reduktion)

Experiment: Huffman-Kompression von Bildern

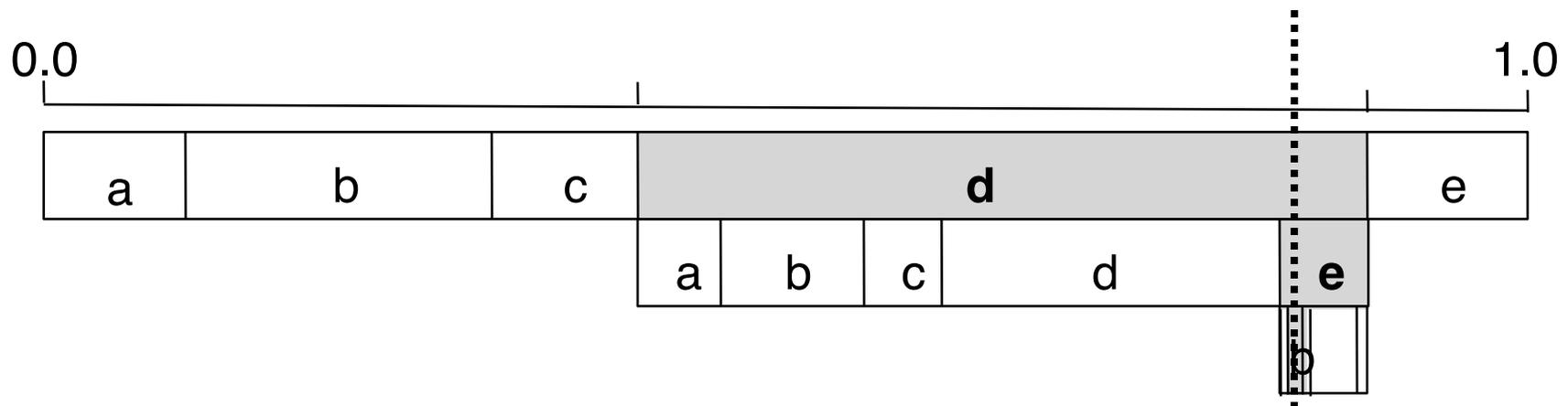
- Grautonbild, 256 x 256 Pixel, 8 bit (d.h. 256 Graustufen)
- Unkomprimiert: 65.536 Bytes
- Mit Huffman kodiert: 40.543 Bytes ca. 38% Reduktion
- Einfacher "Zusatztrick":
 - *Differenz* zwischen benachbarten Pixeln speichern und Huffman dann anwenden
33.880 Bytes ca. 51% Reduktion
 - Keine universelle Kompression mehr, sondern speziell für Pixelbilder
 - Solche "semantischen Kodierungen" siehe später!

Kompressionsverfahren: Übersicht

- Klassifikationen:
 - Universell vs. speziell (für bestimmte Informationstypen)
 - Verlustfrei vs. verlustbehaftet
 - In diesem Kapitel: nur universelle & verlustfreie Verfahren
- Im folgenden vorgestellte Verfahren:
 - Statistische Verfahren:
 - » Huffman-Codierung
 - » Arithmetische Codierung 
 - Zeichenorientierte Verfahren:
 - » Lauflängencodierung (RLE Run Length Encoding)
 - » LZW-Codierung

Arithmetische Codierung (1)

- Gegeben: Zeichenvorrat und Häufigkeitsverteilung
- Ziel: Bessere Eignung für Häufigkeiten, die keine Kehrwerte von Zweierpotenzen sind
- Patentiertes Verfahren; nur mit Lizenz verwendbar
- Grundidee:
 - Code = Gleitkommazahl berechnet aus den Zeichenhäufigkeiten
 - Jedes Eingabezeichen bestimmt ein Teilintervall



Arithmetische Codierung (2)

Beispiel:

Zeichenindex i	1=Leertz.	2=l	3=M	4=S	5=W
Häufigkeit p_i	0.1	0.2	0.1	0.5	0.1
linker Rand L_i	0.0	0.1	0.3	0.4	0.9
rechter Rand R_i	0.1	0.3	0.4	0.9	1.0

Algorithmus:

real L = 0.0; **real** R = 1.0;

Solange Zeichen vorhanden **wiederhole**

Lies Zeichen und bestimme Zeichenindex i;

real B = (R-L);

R = L + B*R_i;

L = L + B*L_i;

Ende Wiederholung;

Code des Textes ist Zahl im Intervall (L, R]

Allgemein:

$$L_i = \sum_{j=1}^{i-1} p_j \quad R_i = \sum_{j=1}^i p_j$$

Algorithmus in
"Pseudocode":

"**real**" Datentyp
(Gleitkommazahl)

"=" Zuweisung an
Variable

Arithmetische Codierung (3)

- Beispieltext-Codierung ("SWISS_MISS")

Zeichen	L	R
	0.0	1.0
S	0,4	0,9
W	0,85	0,9
I	0,855	0,865
S	0,859	0,864
S	0,861	0,8635
Leerz.	0,861	0,86125
M	0,861075	0,86110
I	0,8610775	0,8610782
S	0,86107778	0,86107813
S	0,86107792	0,861078095

real L = 0.0; **real** R = 1.0;

Solange Zeichen vorhanden **wiederhole**

Lies Zeichen und bestimme Zeichenindex i;

real B = (R-L);

R = L + B*R_i;

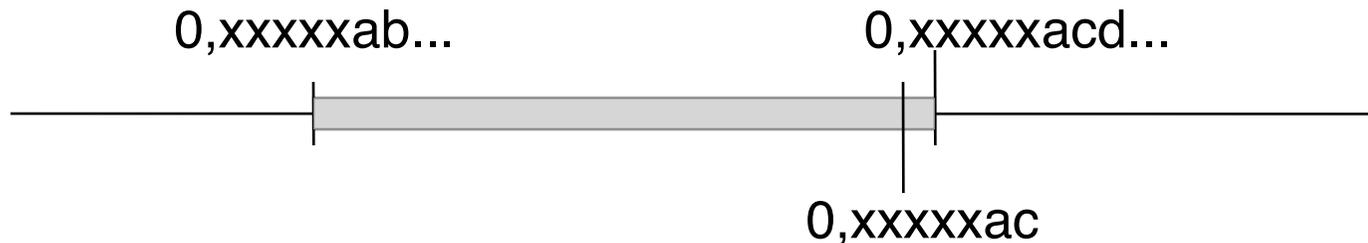
L = L + B*L_i;

Ende Wiederholung;

Zeichenindex i	1=Leerz.	2=I	3=M	4=S	5=W
Häufigkeit p _i	0.1	0.2	0.1	0.5	0.1
linker Rand L _i	0.0	0.1	0.3	0.4	0.9
rechter Rand R _i	0.1	0.3	0.4	0.9	1.0

Arithmetische Kodierung (4)

- Problem Gleitkomma-Arithmetik:
 - Konversion in Ganzzahl-Bereich durch "Skalieren"
- Welcher Binärcode:
 - Ober- und Untergrenze binär codieren
 - Code = Oberer Wert, abgebrochen nach der ersten Stelle, die verschieden vom unteren Wert ist



Kompressionsverfahren: Übersicht

- Klassifikationen:
 - Universell vs. speziell (für bestimmte Informationstypen)
 - Verlustfrei vs. verlustbehaftet
 - In diesem Kapitel: nur universelle & verlustfreie Verfahren
- Im folgenden vorgestellte Verfahren:
 - Statistische Verfahren:
 - » Huffman-Codierung
 - » Arithmetische Codierung
 - Zeichenorientierte Verfahren:
 - » Lauflängencodierung (RLE Run Length Encoding) 
 - » LZW-Codierung

Laufängencodierung

- Unkomprimierte Repräsentationen von Information enthalten häufig Wiederholungen desselben Zeichens (z.B. lange Folgen von x00- oder xFF-Bytes)
- Idee: Ersetzen einer Folge gleicher Zeichen durch 1 Zeichen + Zähler
- Eingesetzt z.B. in Fax-Standards

- Beispiel:
aaaabcdeeeffgggghiabtttiikkddde
ersetzt durch
#a4bcd#e3f#g4hiab#t3#i2#k3#d3e

- Probleme:
 - Bei geringer Häufigkeit von Wiederholungen ineffektiv (verschlechternd)
 - Syntaktische Trennung von Wiederholungsindikatoren und unverändertem Code

Kompressionsverfahren: Übersicht

- Klassifikationen:
 - Universell vs. speziell (für bestimmte Informationstypen)
 - Verlustfrei vs. verlustbehaftet
 - In diesem Kapitel: nur universelle & verlustfreie Verfahren
- Im folgenden vorgestellte Verfahren:
 - Statistische Verfahren:
 - » Huffman-Codierung
 - » Arithmetische Codierung
 - Zeichenorientierte Verfahren:
 - » Lauflängencodierung (RLE Run Length Encoding)
 - » LZW-Codierung 

Wörterbuch-Kompressionen

- Grundidee:
 - Suche nach dem „Vokabular“ des Dokuments, d.h. nach sich wiederholenden Teilsequenzen
 - Erstelle Tabelle: Index --> Teilsequenz („Wort“)
 - Tabelle wird dynamisch während der Kodierung aufgebaut
 - Codiere Original als Folge von Indizes
- Praktische Algorithmen:
 - Abraham Lempel, Jacob Ziv (Israel), Ende 70er-Jahre
 - » LZ77- und LZ78-Algorithmen
 - Verbessert 1984 von A. Welch = „LZW“-Algorithmus (Lempel/Ziv/Welch)
 - Basis vieler semantikunabhängiger Kompressionsverfahren (z.B. UNIX „compress“, Zip, gzip, V42.bis)
 - Verwendet in vielen Multimedia-Datenformaten (z.B. GIF)

Prinzip der LZW-Codierung

- Nicht alle Teilworte ins Wörterbuch, sondern nur eine "Kette" von Teilworten, die sich um je ein Zeichen überschneiden.
- Sequentieller Aufbau:
Neu einzutragendes Teilwort = Kürzestes ("erstes") noch nicht eingetragenes Teilwort
- Beispiel:

b a n a n e n a n b a u

ba an na ane en nan nb bau

- Codierung:

b a n a n e n a n b a u

Neu ins Wörterbuch einzutragen, codiert nach altem Wb.-Zustand

LZW-Codierung (1)

- Tabelle mit Abbildung Zeichenreihe -> Indizes
 - Vorbesetzung der Tabelle mit fest vereinbarten Codes für Einzelzeichen (muß nicht explizit gespeichert und übertragen werden)

- Prinzipieller Ablauf:

SeqChar $p = \langle \text{NächstesEingabezeichen} \rangle$;

Char $k = \text{NächstesEingabezeichen}$;

Wiederhole:

Falls $p \ \& \ \langle k \rangle$ in Tabelle enthalten

dann $p = p \ \& \ \langle k \rangle$

sonst trage $p \ \& \ \langle k \rangle$ neu in Tabelle ein

(und erzeuge neuen Index dafür);

Schreibe Tabellenindex von p auf Ausgabe;

$p = \langle k \rangle$;

Ende Fallunterscheidung;

$k = \text{NächstesEingabezeichen}$;

solange bis Eingabeende

Schreibe Tabellenindex von p auf Ausgabe;

Algorithmus-Beschreibung (“Pseudo-Code”)

- Variablen (ähnlich zu C/Java-Syntax):
 - Datentyp fett geschrieben, gefolgt vom Namen der Variablen
 - Zuweisung an Variable mit “=”
- Datentypen:
 - **int**: Ganze Zahlen
 - **Char**: Zeichen (Buchstaben, Zahlen, Sonderzeichen)
 - **SeqChar**: Zeichenreihen (Sequenzen von Zeichen)
 - » Einelementige Zeichenreihe aus einem Zeichen: < x >
 - » Aneinanderreihung (Konkatenation) mit &
- NächstesEingabezeichen:
 - Liefert nächstes Zeichen der Eingabe und schaltet Leseposition im Eingabepuffer um ein Zeichen weiter

LZW-Codierung (2)

- Vorbesetzte Tabelle (z.B. mit ASCII-Codes):
[(`<a>`, 97), (``, 98), (`<c>`, 99), (`<d>`, 100), (`<e>`, 101), (`<f>`, 102), (`<g>`, 103),
(`<h>`, 104), (`<i>`, 105), (`<j>`, 106), (`<k>`, 107), (`<l>`, 108), (`<m>`, 109),
(`<n>`, 110), (`<o>`, 111), (`<p>`, 112), (`<q>`, 113), (`<r>`, 114), (`<s>`, 115),
(`<t>`, 116), (`<u>`, 117), (`<v>`, 118), (`<w>`, 119), (`<x>`, 120), (`<y>`, 121),
(`<z>`, 122)]
- Für neue Einträge z.B. Nummern von 256 aufwärts verwendet.

LZW-Codierung (3)

Beispieltext: "bananenbau"

Ablauf:

Lesen (k)	Codetabelle schreiben (p & <k>)	Ausgabe	Puffer füllen (p)
			
a	(<ba>, 256)	98	<a>
n	(<an>, 257)	97	<n>
a	(<na>, 258)	110	<a>
n			<an>
e	(<ane>, 259)	257	<e>
n	(<en>, 260)	101	<n>
a			<na>
n	(<nan>, 261)	258	<n>
b	(<nb>, 262)	110	
a			<ba>
u	(<bau>, 263)	256	<u>
EOF		117	

LZW-Decodierung bei bekannter Tabelle

Wiederhole solange Eingabe nicht leer:

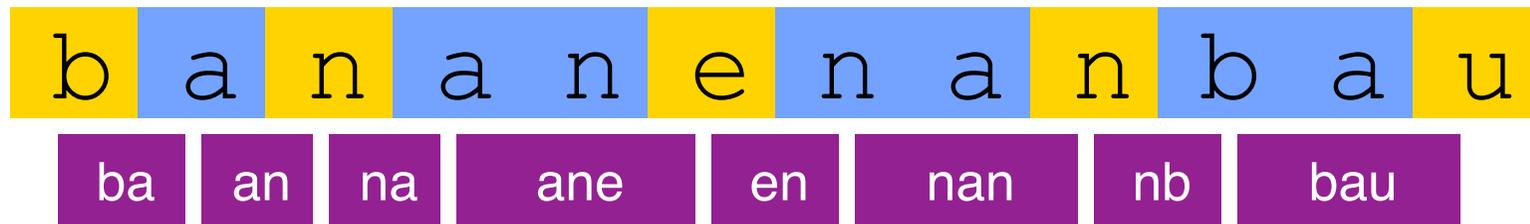
k = NächsteEingabezahl;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

Ende Wiederholung;

LZW-Decodierung (1)

- Grundidee („symmetrische Codierung“):
 - Das aufgebaute Wörterbuch muß *nicht* zum Empfänger übertragen werden.
 - Das Wörterbuch wird nach dem gleichen Prinzip wie bei der Codierung bei der Decodierung dynamisch aufgebaut.
 - Das funktioniert, weil bei der Codierung immer *zuerst* der neue Eintrag für das Wörterbuch nach bekannten Regeln aus dem schon gelesenen Text aufgebaut wird, bevor der neue Eintrag in der Ausgabe verwendet wird.
- Algorithmusidee:
 - Neu einzutragendes Teilwort = letztes Teilwort plus erstes Zeichen des aktuellen Teilworts



LZW-Decodierung (2)

Prinzipieller Algorithmus:

SeqChar $p := \langle \rangle$;

int $k = \text{NächsteEingabezahl}$;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

int $old = k$;

Wiederhole solange Eingabe nicht leer:

$k = \text{NächsteEingabezahl}$;

SeqChar $akt = \text{Zeichenreihe mit Tabellenindex } k$;

Schreibe Zeichenreihe akt auf Ausgabe;

$p = \text{Zeichenreihe mit Tabellenindex } old \text{ (letztes Teilwort)}$;

Char $q = \text{erstes Zeichen von } akt$;

Trage $p \ \& \ \langle q \rangle$ in Tabelle ein
(und erzeuge neuen Index dafür);

$old = k$;

Ende Wiederholung;

LZW-Decodierung (3)

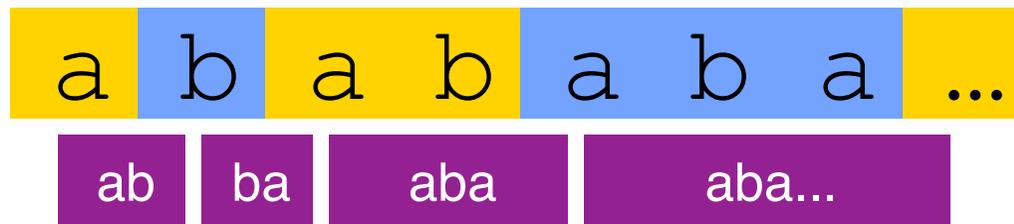
- Beispielzeichenreihe: "98-97-110-257-101-258-110-256-117"
- Ablauf:

Lesen (k)	Ausgabe (q ist jeweils <u>unterstrichen</u>)	Puffer füllen (p)	Codetabelle schreiben (p & <q>)	Merken (old)
98	b			98
97	<u>a</u>	b	(<ba>, 256)	97
110	<u>n</u>	a	(<an>, 257)	110
257	<u>an</u>	n	(<na>, 258)	257
101	<u>e</u>	an	(<ane>, 259)	101
258	<u>na</u>	e	(<en>, 260)	258
110	<u>n</u>	na	(<nan>, 261)	110
256	<u>ba</u>	n	(<nb>, 262)	256
117	<u>u</u>	ba	(<bau>, 263)	117
EOF				

LZW-Decodierung (4)

- Beispielzeichenreihe: “abababa...“, Beispielcode: “97-98-256-258“
- Ablauf:

Lesen (k)	Ausgabe (q ist jeweils unterstrichen)	Puffer füllen (p)	Codetabelle schreiben (p & <q>)	Merken (old)
97	a			97
98	<u>b</u>	a	(<ab>, 256)	98
256	<u>ab</u>	b	(<ba>, 257)	256
258	???			



Decodierung ist so noch nicht korrekt!

LZW-Decodierung, vollständige Fassung

SeqChar $p := \langle \rangle$;

int $k =$ NächsteEingabezahl;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

int $old = k$;

Wiederhole solange Eingabe nicht leer:

$k =$ NächsteEingabezahl;

SeqChar $akt =$ Zeichenreihe mit Tabellenindex k ;

$p =$ Zeichenreihe mit Tabellenindex old (letztes Teilwort);

Falls Index k in Tabelle enthalten

dann **Char** $q =$ erstes Zeichen von akt ;

Schreibe Zeichenreihe akt auf Ausgabe;

sonst **Char** $q =$ erstes Zeichen von p ;

Schreibe Zeichenreihe $p \ \& \ \langle q \rangle$ auf Ausgabe;

Ende Fallunterscheidung;

Trage $p \ \& \ \langle q \rangle$ in Tabelle ein
(und erzeuge neuen Index dafür);

$old = k$;

Ende Wiederholung;