3.8 Teilbandcodierung

3.8.1 Perfekte Teilbandrekonstruktion

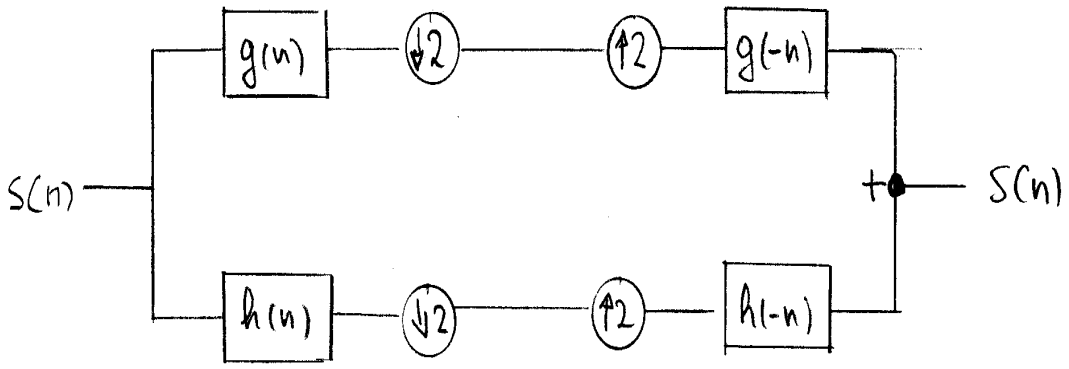
Damit das Originalsignal trotz Zerlegung/Summation *ohne Verzerrung* zurückgewonnen wird, benutzt man

Filter mit (fast) perfekter Rekonstruktionseigenschaft (Englisch PRF Filter)

Diese braucht man überall, wo Subband-Coding vorkommt: Sprach/Audio/Bildcodiernung.

Für zwei Frequenzbänder mit normierter Abtastfrequenz 1:

- g(n): Tiefpass mit dazu g(-n) gespiegelter Tiefpass
- h(n): Hochpass mit dazu h(-n) gespiegelter Hochpass



Mit speziellen Filtern fehlerfrei trotz Unterabtastung!
Es gibt dafür sog. QMF-Filter (Quadrature Mirror Filter). Für diese gilt noch speziell

(3.1) $h(n) = (-1)^n g(N-1-n),\ n = 0, 1, ...N-1$

d.h. der Hochpass wird aus dem Tiefpass mit alternierendem Vorzeichen hergeleitet, d.h. man braucht nur die Tiefpasskoeffizienten.

- Populäre brauchbare Koeffizientensätze in Tabelle
- Es gibt noch vieles mehr, Forschungsthema
- z. B. Smith-Barnwell Koeffizienten mit mehr Trennschärfe

Aus 2-Kanal Bank lässt sich durch Kaskadierung 2^n Kanal Filterbank mit perfekter Rekonstruktion aufbauen, siehe die ausgegebenen Unterlagen.

Es gibt aber auch *direkte Konstruktion* spezieller Filter für M-Kanal Bänke mit perfekter Rekonstruktion

z.B für 32 Bänder bei MPEG 1-Audio Layer 1 oder 2

3.8.2 Teilbandcodierung/Kompression

Wie kann man damit komprimieren?

Möglichkeiten:

- In jedem Teilband DPCM oder ADPCM mit weniger Bits
- Oder optimale Bitzuweisung (bit allocation) für die Teilbänder

Idee: Teilbänder mit viel Energie bekommen viele Bits, die mit wenig Energie wenig Bits

Signalenergie:

$$E_s = \sum_{n=1}^{N} |s(n)|^2$$

(Energie eines endlichen Signalausschnitts)

Es gibt eine Formel für die optimale Bitzuweisung, die ist in der Praxis nicht so recht brauchbar, weil negative Bitanzahlen entstehen können.

Stattdessen guter, heuristischer Algorithmus: Sei

R Anzahl Bits/Sample, die angestrebt wird (vorgegeben!)
M Anzahl Teilbänder (Datenrate 1/M tel)
Für jedes Teilband im Mittel R Bits, also insgesamt für M Teilbänder
R_b = MR Bits, die geschickt verteilt werden müssen. Sei
R_k Anzahl Bits für Teilband k
E_k Energie im Teilband k

Verfahren: Bilde Paar (E_k, R_k) für jedes Teilband

1. Initialisierung: Setze $R_k = 0$ für alle k, $R_b = MR$
2. Suche maximales E_k, bilde dort $(\frac{E_k}{2}, R_k + 1)$ (Energie halbieren, ein Bit dazu)
3. Bilde $R_b = R_b - 1$ (Bitreserve um 1 vermindern); falls $R_b = 0$, anhalten, sonst weiter mit 2)

Nach endlich vielen Schritten sind alle Bits verteilt, die Bänder mit viel Energie haben viele Bits (und damit einen besseren SNRQ von $6R_k$dB), die anderen wenig.

Das funktioniert gut (deterministisch, schnell, einfach)

Die Teibänder werden dann mit R_k Bit quantisiert (linear oder nichtlinear), dann übertragen/gespeichert und rekonstruiert. Natürlich muss dem Decoder mitgeteilt werden, wieviel Bit jedes Teilband hatte, damit korrekt decodiert (epandiert auf R Bit pro Band) werden kann.

Zu beachten:

- Die Quantisierung hat so zu erfolgen, dass jedes Teilband voll ausgesteuert ist. (Grund: S/N_q bezieht sich auf den voll ausgesteuerten Fall). Daher muss jedes Teilband vor der Quantisierung skaliert werden (durch seinen maximalen Wert geteilt werden), und zum Rekonstruieren muss die Skalierung rückgängig gemacht werden. Die Skalierungsfaktoren müssen also ebenfalls mitgeschleppt werden:

- Da Audiodaten ihre Charakteristik mit der Zeit ändern, ist es sinnvoll, die Werte in den Teilbändern in Frames (Gruppen von Werten) einzuteilen und in jedem Frame Bitzuweisung und Skalierung immer wieder neu vorzunehmen. Nachteil:
  - Rechenzeit
  - Mehr Seiteninformation

Lohnt sich aber trotzdem. Typische Framelänge: 8 msec

Funktioniert gut, aber noch nicht gut genug. Es geht noch besser mit psychoakustischer Codierung

3.8.3 Psychoakustische Codierung/ perceptual coding

Hierfür Teilbandzerlegung nötig

- Verlustbehaftete Audiocodierung
- Kompression auf 2 Bits/Sample bei CD-Qualität

Prinzip: **man lässt weg, was man nicht hört.** Es gibt

- frequenzabhängige Ruhegehörschwelle, darunter hört man nichts mehr
- Maskierungseffekte, laute Töne verdecken leise

Technische Konsequenz:

- Signale unterhalb der Gehörschwelle können weggelassen werden
- Man darf die Quantisierung so grob vornehmen, dass das Quantisierungsrauschen knapp unter der Gehörschwelle bleibt (Bits sparen!)

Etwas genauer: Untersuchungen des menschlichen Gehörs:

Innerhalb bestimmter Frequenzbänder (sog. kritische Bänder) verdecken laute Signale leise besonders stark, in benachbarten Bändern wird der Effekt schwächer. Dies ist auch noch von der Lautstärke abhängig.

Ein lautes Signal erzeugt also einen *Masking Threshold* (Maskierungsschwelle), unterhalb derer nichts mehr hörbar ist.

Die genauen Maskierungskurven werden in den diversen Implementierungen durch einfache Kurven angenähert.

3.8.4 MPEG 1 Layer 1, 2 und 3 (MP3)

Layer 1 und 2:

- MPEG 1 Layer 1 und 2:
  - 32-Kanal Filterbank mit PRF-Filtern
  - 1024-Punkte FFT liefert Daten für das psychoakustische Modell, Identifizierung der lauten Signale in jedem Frequenzband
  - Lineare Quantisierung, Bitanzahl pro Frequenzband wird durch das psychoakustische Modell geliefert
  - 32-192 kbit/sec, höchste Datenrate liefert CD-Qualität, die bräuchte sonst 705 kbit/sec
- MPEG Layer 3 (MP3)
  - 32 Bänder nicht fein genug für kritische Bänder
  - Daher weitere Frequenzzerlegung mit MDCT (18 Werte) auf 40 Hz Bandbreite in jedem der 32 Teilbänder
  - MDCT: Modified Discrete Cosine Transform, Verwandte der DFT, Frequenzzerlegung in reelle Werte mit verlustfreier inverser MDCT
  - Bitzuweisung iterativ: In jedem Band, das codiert wird, gibt es einen Noise to Masking Ratio (NMR) in dB. Die zu Verfügung stehenden Bits werden solange auf die Bänder mit dem jeweils schlechtesten NMR verteilt, bis keine mehr da sind
  - Danach adaptive statische Huffman-Codierung (nicht bei Layer 1 und 2), der Encoder kann aus 32 statischen Huffman-Codes den besten auswählen

Ergebnis MP3: 128 kbit/s bei guter Qualität

Warum gibt es noch andere (bessere) Verfahren?

MP3 nutzt die ebenfalls vorhandene zeitliche Maskierung nicht aus. Ein lautes Signal kann ein leises, das zeitlich kurz vorher oder nachher konmmt, verdecken. Vor und nach dem lauten Signal kann also auch mehr Quantisierungsrauschen erlaubt sein.

Wird benutzt bei MPEG 2 AAC (Adanced Audio Coding), besser als MP3, aber nicht kompatibel zu MPEG 1.

MPEG: Moving Pictures Expert Group, von manchen auch als Moving Experts Picture Group bezeichnet.

Kapitel 4

Verlustbehaftete Bildkompression

Wichtiges Thema:
Computergrafik, Spiele, Videos, Digitalkameras produzieren immer grössere Mengen an Bilddaten.

Altes Schwarzweissbild: 512x512x8 Bit = 256 kByte
Neueres Farbbild: 1024x1024x24 Bit= 3 MByte

Bemerkung

- 8 Bit Quantisierung pro Farbebene ausreichend, ausser für Spezialanwendungen wie Röntgenbilder, dort 12-16 Bit
- Digitale Fernsehen 24 Bilder/sec

Also: Kompression tut not!

Es wurde viel getan für verlustfreie Verfahren (Basis LZ77, LZ78, Huffman etc)
Je nach Bildtyp Faktor 1.1 - 10
Bekannte Formate mit verlustfreier Kompression:
PCX, GIF, BMP, PNG

Kompressionsfaktoren bescheiden, denn:

- Bei photographischen Bildern sind die Wahrscheinlichkeiten der einzelnen Pixelintensitäten relativ gleich (Messergebnisse), d. h. Huffman und arithmetische Codierung können nicht viel bringen
- Bestimmte Muster oder Pixelfolgen wiederholen sich subjektiv (Auge) oft (Ziegelsteine, Dachpfannen ...), digitalisiert gibt es aber leichte Abweichungen, die das Auge nicht wahrnimmt

  Für lexikalische Kompression sind diese kleinen Abweichungen aber fatal, da Symbolfolgen exakt übereinstimmen müssen

Verbesserungsversuche

- 1-2D Differenzenbilder codieren, leichte Verbesserung für wenig Aufwand
- 2-D Prädiktion, etwas besser mit mehr Aufwand
- Faktor 2 bei Naturbildern ist ungefähr die Grenze

4.1 Verlustbehaftete Verfahren

Wie bei Sprache:

- Leichte Änderungen für bessere Kompression sind (fast) nicht wahrnehmbar
- Die meisten Bilder sind Fotografien, eingescannte Dokumente etc., also bereits durch Aufnahme verzerrte Abbbilder der realen Umwelt. Daher sind kleine Änderungen nicht wahrnehmbar

4.1.1 Einfachste Ansätze

- Quantisierung gröber:
  Von 256 Graustufen auf 64 mit nichtlinearer Kennlinie fast ohne Qualitätseinbußen (lineare Kennlinie funktioniert auch schon ganz gut)
- Abtastrate heruntersetzten (nach Tiefpassfilterung):
  Oft erstaunlich gut

4.2 JPEG-Codierung

JPEG: Joint Photographic Experts Group

- Standard-Verfahren (hat lange gedauert) von 1991
- Verlustfreie und verlustbehaftete Kompression
- Faktor 8 ohne große Verluste an Qualität, d.h. nicht wahrnehmbar

Hier: Verlustbehafteter Teil

Prinzip:

JPEG ist eine Transformationscodierung
Basis ist die DCT - Discrete Cosine Transform

Bemerkung:
Es gibt viele Transformationscodierungen. Man benutzt

- DFT Discrete Fourier Transform
- DHT Discrete Haar Transform (schlechter als DCT)
- DWT Discrete Wavelet Transform (neuer, besser als DCT)

DCT ähnlich wie DFT

Vorteile:

- Besser zur Kompression geeignet (empirisches Ergebnis, hat man in den 1970ern untersucht)
- Nur reelle Koeffizienten

- Schneller Algorithmus (den braucht jede Transformationscodierung!), N Werte mit $\approx N \log_2 N$ Rechenschritten transformiert

Die Koeffizienten, die von der DCT geliefert werden, können wie bei der DFT als Frequenzanteile interpretiert werden

Definition
Sei s(n,m) eine (Bild)Matrix, $0 \leq n \leq N-1$, $0 \leq m \leq N-1$ mit N^2 Elementen. Die DCT von s(n,m) ist gegeben durch

$$DCT(u,v) = \frac{1}{\sqrt{2}N} C(u)C(v) \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s(n,m) \cos(\frac{(2m+1)u\pi}{2N}) \cos(\frac{(2n+1)v\pi}{2N})$$

mit $0 \leq u \leq N-1$ und $0 \leq v \leq N-1$.

Die Konstanten C(u), C(v) sind definiert durch

$$\begin{aligned} C(u) &= \frac{1}{\sqrt{2}} \text{ für } u = 0, \text{, sonst } C(u) = 1 \\ C(v) &= \frac{1}{\sqrt{2}} \text{ für } v = 0, \text{, sonst } C(v) = 1 \end{aligned}$$

mit $0 \leq u \leq N-1$ und $0 \leq v \leq N-1$.
DCT(u,v) ist wieder eine NxN Matrix von DCT-Koeffizienten. Aus diesen kann man das Original fehlerfrei zurückgewinnen mittels IDCT = Inverse DCT

$$\begin{aligned} S(n,m) = IDCT(n,m) &= \frac{1}{\sqrt{2N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)DCT(u,v) \\ &\quad \times \cos(\frac{(2n+1)u\pi}{2N}) \cos(\frac{(2m+1)v\pi}{2N}) \end{aligned}$$

mit $0 \leq n \leq N-1$ und $0 \leq m \leq M-1$.

Die Konstanten C(u), C(v) sind definiert wie oben.

Diese direkten Formeln sind leicht programmierbar, der schnelle Algorithmus leider nicht.

MATLAB-Beispiel:
load lisa; lisadct=dct2(lisa);colormap(jet(64)); imagesc(log(abs(lisadct)));

Frage
Was soll das?
DCT führt NxN Bildmatrix in NxN Koeffizientenmatrix über
Kein Gewinn für Kompression (N/10 x N/10 Matrix wäre besser)

Betrachte Bild einer DCT-Transformierten:

Oben links sitzen die tiefen Frequenzen; nach unten rechts werden die Frequenzen immer höher
Bei realen Bildern werden die Koeffizienten mit höheren Frequenzen schnell sehr klein und können oft vernachlässigt werden. Die DCT packt die wichtigen Informationen nach oben links hin. Das kann man zur Kompression ausnutzen.

Am Originalbild ist das nicht zu machen, daher Wichtigkeit der Transformation.

JPEG implementiert, wie man welche Information weglässt!

4.2.1 Blockcodierung

DCT für z.B. 256x256 Bild braucht schon recht lange. Zur Bildcodierung wird das Bild unterteilt in 8x8 Blöcke, die einzeln transformiert werden.

Forschungsergebnisse

- Blockgrößen von mehr als 64x64 bringen keinen Gewinn
- 16x16 ist fast so gut wie 64x64
- Owohl 8x8 Blöcke schlechter als 16x16, hat JPEG 8x8 gewählt
- Einziger Grund: Rechenzeit (1991!)

Bemerkung

- JPEG mit 16x16 Blöcken wäre besser und leicht zu implementieren (lohnt sich nicht mehr)
- Alle Codierungsverfahren, die Bilder in Blöcke aufteilen, werden als Blockcodierung bezeichet (DCT, DFT, DHT,...)
- Wavelet-Codierung ist keine Blockcodierung

4.2.2 Quantisierung

DCT ist an sich verlustfrei. Beim Rechnen mit ganzen Zahlen nimmt die Koeffizientenmatrix sogar mehr Speicher ein als das Original!
Ausgehend von 8 Bit Pixeln in 8x8 Block können die DCT-Koeffizienten im Wertebereich

(-1024,1023)

liegen, benötigen also 11 Bit (Expansion!)

Was tun?
Erneute Quantisierung mit speziellem Verfahren zur Reduktion der Bitanzahl!

Idee: Je höher die Frequenz = Abstand vom DC Wert (Gleichanteil), umso geringer die Bitzahl. Es gibt dafür einige Möglichkeiten.

JPEG benutzt Quantisierungsmatrizen. Das sind 8x8 Matrizen mit Quantisierungswerten q(n,m). Die Werte von q(n,m) werden mit höheren Indizes größer.

JPEG-Quantisierungsformel (einfach)

$$\text{Quantcoeff}(n,m) = round(\frac{DCT(n,m)}{q(n,m)})$$

Große Werte von q(n,m) bedeuten, dass der Koeffizient auf 0 gerundet wird.
Decoder berechnet natürlich

$$DCT(n,m) \approx \text{Quantcoeff}(n,m) * q(n,m)$$

Wahl einer Quantisierungsmatrix:
Jeder kann seine eigene machen! JPEG schlägt verschiedene vor, die mit Qualitätsfaktoren versehen sind

Kriterien zu Wahl von q(n,m):

1. Objektives Kriterium: MSE und PSNR

   - MSE (Mean Square Error), mittlerer quadratischer Fehler:
     s(n,m) Originalbild, $s_d(n,m)$ decodiertes Bild

     $$MSE = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} (s(n.m) - s_d(n,m))^2$$

   - PSNR (Peak Signal to Noise Ratio)
     Für die Bewertung von Bildern üblich. Es dient als Referenz der Spitzenwert s_{max} zum Quadrat, also bei 8 Bit Bildern 255^2:

     $$PSNR = 10 \log_{10}(\frac{s_{max}^2}{MSE}) \; [dB]$$

2. Subjektives Kriterium: Visuelle Qualität
   PSNR ist kein dem menschlichen Auge angepasstes Kriterium. Daher müssen die Quantisierungsmatritzen von Menschen beurteilt werden. Dazu wurden lange Versuchsserien mit Beobachtern durchgeführt

Es gibt natürlich gute und schlechte Quantisierungsmatritzen. Eine der Hauptaufaben von JPEG war, **Optimale!?** Quantisierungsmatritzen zu finden.

4.2.3 Verlustfreie Nachbearbeitung

Es liegen nunmehr die quantisierten Koeffizienten Quantcoeff(n,m)=QC(n,m) vor. Bei der Quantisierung (und nur dort) werden die Verluste gemacht.

Die weitere Verarbeitung ist noch ziemlich ausgetüfelt. Es hat sich gezeigt, dass die DC-Koeffizienten = QC(0,0) separat behandelt werden sollten, um besser zu komprimieren. Grund:

Die QC(0,0) Koeffizienten der einzelnen Blöcke hängen stark zusammen, stärker als mit dem Rest der 8x8 Koeffizienten.
Daher

1. Schritt: die DC-Koeffizienten werden differenzencodiert (in der unten beschriebenen Zick-Zack Reihenfolge) und die Differenzen werden dann Huffman-codiert (mit einem festen Huffmancode, in Codierer und Decodierer bekannt)

2. Schritt: die restlichen (AC) Koffizienten der einzelnen Blöcke werden in Zick-Zack Richtung ausgelesen (nicht Zeilen- oder Spaltenweise)
   Grund: Höhere Korrelation, da der nächste Pixel der Folge geometrisch in der Nähe des Vorgängers liegt, also ähnliche Werte aufweist. Die Koeffizientenfolge wird dann lauflängencodiert:
   Anzahl 0-en Wert Anzahl 0-en Wert ...
   Da sehr viele 0-en Vorkommen, komprimiert das gut.
   Die Ergebnisse der Lauflängencodierung werden dann wieder Huffman-codiert, mit einem festen Code, aber einem *anderen* als für die DC-Koeffizienten

4.2.4 Nachteile von JPEG

1. Wie für alle Blockcodierungsverfahren: Bei höheren Kompressionsfaktoren sieht man die Blöcke - sehr störend
2. statische Huffmancodierung ist schlechter als adaptive Huffman oder arithmetische Codierung
3. Blockgröße ist nicht optimal

Lösung für 2), 3) mit einem verbesserten JPEG möglich. Dies wurde von Eric Dietz in seiner Diplomarbeit konkret nachgewiesen.

Eine zufriedenstellende Lösung für 1) gibt es nicht. Blockeffekte vermeidet die Wavelet-Transformationskompression, die auf einer Multiscalenzerlegung des ganzen Bildes beruht, ähnlich wie eine 2D-Subband-zerlegung.

JPEG2000 basiert auf der Wavelet-Transformation und liefert bessere Kompressionsfaktoren als JPEG (aber nicht viel besser als ein nach 2), 3) verbessertes JPEG...) Dafür ist der Speicherbedarf gewaltig (Riesenprogramm), d.h für viele Anwendungen ist JPEG2000 nicht geeignet.