Bild/Video-Quellencodierung

JPEG Intraframecodierung

- Bilder werden aufgeschnitten (z.B. in 8x8 Blöcke) → DCT (horizontal und vertikal) → Quantisierung → Redundanzreduktion
- Nach der DCT könnte man das Bild schon (bis auf Rundungsfehler) verlustfrei wiederherstellen → IDCT
  - ➔ Umgekehrte Reihenfolge der Schritte für Decodierung (Grundprinzip der Nachrichtentechnik)

DCT (Diskrete Cosinus Transformation) und Redundanzreduktion

- Der Cosinus wird verwendet, um imaginäre Anteile bei der Transformation zu vermeiden
- DCT-Formel:

$$S(X,Y)=\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}\alpha(X)\cdot\alpha(Y)\cdot s(x,y)\cdot\cos\left(\frac{(2x+1)\cdot X\cdot\pi}{2N}\right)\cdot\cos\left(\frac{(2y+1)\cdot Y\cdot\pi}{2M}\right)$$

  $s(x,y)$ = Bildsignal im Originalbereich mit den Koordinaten x, y

  $S(X,Y)$ = Bildsignal im Ortsfrequenzbereich mit den Koordinaten X, Y

  N,M = Anzahl Zeilen und Spalten der Matrix

  α = Gewichtungskoeffizient

$$\alpha(r)=\begin{cases}\sqrt{\frac{1}{N}} & \text{für } r=0\\ \sqrt{\frac{2}{N}} & \text{für } r\neq 0\end{cases}$$

- Bsp. für eine Matrix von 8x8 Werten

$$S(X,Y)=\frac{1}{4}\cdot\beta(X)\cdot\beta(Y)\cdot\sum_{x=0}^{7}\sum_{y=0}^{7}s(x,y)\cdot\cos\left(\frac{(2x+1)\cdot X\cdot\pi}{16}\right)\cdot\cos\left(\frac{(2y+1)\cdot Y\cdot\pi}{16}\right)$$

  mit

$$\beta(r)=\begin{cases}\frac{1}{\sqrt{2}} & \text{für } r=0\\ 1 & \text{für } r\neq 0\end{cases}$$

- Basisfunktionen:
  - Man erhält nach der DCT hohe Werte in jenen Feldern, in denen entsprechend viele Anteile der Basisfunktion vorliegen
  - Betrachtet man nur die erste Matrix, die gleichhell ist, so gibt diese Auskunft über einen im Originalbildanteil enthaltenen DC-Anteil
  - Je komplexer der Bildinhalt ist, desto weiter wandert man nach unten rechts zu der „wildesten“ Basisfunktion
  - Überlagerung von horizontalen und vertikalen Anteilen → verschachtelte Basisfunktionen
  - Aufwändig im gesamten Ablauf der Codierung ist die Berechnung der DCT
  - Die Phase geht bei der DCT nicht verloren

- Bsp. in Matlab → Bild mit Grashalmen:
    - Gräser sorgen für viel Aktion im Bild → greift man eine Zeile heraus und analysiert diese, so erhält man ein mit DC behaftetes Zufallssignal
    - Transformiert man ein Zufallssignal, so erhält man wieder ein Zufallssignal → alle Koeffizienten sind beteiligt
    - Damit die DCT gut funktioniert, sollte nicht zu viel Willkür im Bild sein
    - Betrachtet man das Bild in 8x8 Blöcken, so sind Muster (→ DCT Basisfunktionen) erkennbar
    - ➔ Die Muster müssen erkennbar sein, weil sonst das Verfahren nicht funktionieren würde
        - Einzelne Ausreißer wären im Zeitverlauf ein Diracstoß → Singularität (alle Frequenzen sind enthalten)
    - ➔ Wenn von Schärfe im Bild die Rede sein soll, dann müssen alle Frequenzen beteiligt sein
    - ➔ Es wird nur jenes herausgegriffen, was auf den ersten Blick hin wichtig ist → Singularitäten können herausgefiltert werden
    - ➔ Verfahren ist z.B. in der Medizintechnik nicht anwendbar → Datenvernichtung
- Bsp. Bildblock, welcher als Matrix (Luminanzwerte) vorliegt:
    - Horizontal liegt kaum etwas an Unterschieden vor
    - In der Bildmitte gibt es eine recht helle Zeile → im horizontalen keine großen Sprünge, im Vertikalen braucht man einige Frequenzen, um diesen Verlauf darstellen zu können
    - Nach der DCT tauchen große Werte in der linken Spalte auf (vertikale Frequenzen)
    - Nun liegen immer noch recht viele und auch hohe Werte vor → man möchte durch Gewichtung der Ergebnismatrix möglichst kleine Werte loswerden
    - ➔ Begründung: Sieht man sich einen Film an, achtet man nicht sofort auf Details und vergisst diese recht schnell → daher wird an dieser Stelle eingespart
        - Gewichtungsmatrix wird mit elementweiser Division angewendet → es entstehen viele Nullen, ansonsten bleiben nur noch kleine Werte übrig
        - Danach wird die Matrix diagonal ausgelesen, wobei man die Nullen mit den anderen Werten paarweise zusammenfasst
        - Auf Grund der sehr oft vorkommenden Null kann eine VLC (Variable Längencodierung, z.B. Huffmann) weiter optimieren, eine DPCM sorgt im DC-Koeffizient für eine weitere Optimierung
    - ➔ Problem bei der DPCM → „Gedächtnis“ → wird ein falscher Wert ermittelt, setzt sich dieser über den gesamten Block fort → dies wird in einer Helligkeitsverschiebung sichtbar

<u>Decodierung</u>

- Entnormieren (elementweise Multiplikation mit Gewichtungsmatrix) → viele Anteile werden durch die Nullen, die an ihre Stelle getreten sind, nicht mehr aktiv
    - ➔ Andere Anteile werden verstärkt → Singularitäten werden hervorgehoben
    - ➔ Schwache Anteile verschwinden
- Im Beispiel wird beim DC-Koeffizient eine Abweichung von 4‰ nicht im Bild sichtbar werden
- Die Rücktransformation zeigt, dass sich die Werte im Bild nur wenig geändert haben:
    - Der Bereich um die helle Linie ist heller geworden, die Linie selbst dunkler
    - Die helleren Anteile im oberen Bereich des Blocks haben an Helligkeit verloren

Praxisbeispiel MPEG

- Bsp. scharfe Kanten von Schriftzeichen oder abstrakten Darstellungen → verursachen viele Werte in den Zeilen und Spalten der Matrix nach der DCT
- Wichtig für eine gut funktionierende Komprimierung ist eine Nutzung der oberen Zeile und der linken Spalte → andere Werte sollten möglichst komplett rausfallen
- Der Vorteil der Komprimierung rangiert aus bereits vorhandenen Signalanteilen, welche in der Kompression genutzt werden → In dem Moment, in dem man weiß, wie das Bild aufgebaut ist, hat man den größten Kompressionsgewinn
- Man muss die knappste Form finden (Bsp. Text in einer PowerPoint Präsentation)
  - ➔ Macht man eine Analyse der Blöcke oder kann man ohne schon die Information entnehmen (in diesem Fall den Text lesen)
- Arbeiten mit Bildebenen → was im Hintergrund läuft, darf stärker prädiktiert werden
  - ➔ Codierung sollte stärker an den Generatorprozess herangeführt werden (Bsp. eine Wand im Vorlesungsraum → man kann nicht einfach weiß vorgeben, sondern muss auch feine Strukturen beachten)
  - ➔ Merkmale müssen extrahiert werden (z.B. Grashalme und ihre Krümmung durch den Wind)
  - ➔ Man sollte immer unterscheiden, ob ein Bild von einer Kamera aufgenommen wurde oder von einem Generator erzeugt worden ist

MPEG Video Encoder

- Besitzt Rückkopplungsschleife, welche das komplette Bild wieder herstellt → es wird von jenem Bild ausgegangen, welches auch der Decoder als Prädiktion vorgegeben bekommt
- Kompressor zur Festlegung der maximalen Datenrate für das aktuelle Bild → auf Multiplexer geschaltet, der erneut zum Quantisierer durchschalten kann
  - ➔ Wichtig: Das Format kann anders sein, als vorgesehen → Werte des Encoders können angepasst werden, ein Arbeiten mit Standardwerten muss nicht passieren → der Decoder versteht auch ein von den Einstellungen verschieden codiertes Signal

Kanalcodierung

Warum wird Kanalcodierung genutzt?

- Es gibt keine fehlerfreien Kanäle → auch auf Weitverkehrsstrecken gibt es Fehlerraten von 10^{-12} → bei Voice stört ein Fehler im Gespräch in der Regel nicht, weil man trotzdem noch alles versteht
- Jedes System hat eine andere Physik, Anwendungen haben andere Anforderungen → eine Kanalcodierung muss an das System und die spezifische Anwendung angepasst werden

Prinzip

- Gezieltes Hinzufügen von Redundanz in den Kanal, um Fehler zu finden → Code muss redundant sein, um Fehler finden zu können
- Der Vorrat an Wörtern des Codes muss auch fehlerbehaftete Werte enthalten, damit diese vom Empfänger erkannt werden können → Nur „korrekte“ Werte werden gesendet
- Vorrat an Wörtern (z.B. 3 Bit = 8 Wörter) wird aufgeteilt in korrekte und falsche Wörter → z.B. 4 Wörter sind korrekt, 4 sind falsch (theoretische könnte man die 4 korrekten Wörter

auch mit 2 Bit codieren, das dritte Bit ist in diesem Fall als Bit zur Fehlererkennung hinzugefügt)

- 2 Möglichkeiten, wie der Empfänger auf einen Fehler reagiert:
  - Es wird neu gesendet und wird dann hoffentlich korrekt übertragen
  - Es wird versucht, den Fehler zu korrigieren
- Möglichkeit der Optimierung → nur 2 korrekte Wörter, aber 6 fehlerhafte Wörter → führt zu größerer Distanz zwischen den korrekten Wörtern

  ➔ 2 Fehler könnte auch in einem Rahmen vorliegen → es müssten 3 Fehler passieren, damit 1 Fehler nicht mehr erkannt werden kann
- Codewörter brauchen einen Mindestabstand:

  $d_{\min} > s + t, s \geq t$

  t = korrigierbare Fehler

  s = erkennbare Fehler
- Eine Korrektur ist nur dann möglich, wenn der Abstand zu einem korrekten Wort kleiner als zu allen anderen ist (ein gleicher Abstand zu zwei korrekten Wörtern macht die Korrektur unmöglich)

<u>Übersicht der erkennbaren und korrigierbaren Fehler bei verschiedenen Abständen</u>

$d_{\min}$	t (korrigierbare Fehler)	s (erkennbare Fehler)	Anwendungsbereich
2	0	1	
3	0	2	ARQ
3	1	1	FEC
4	0	3	ARQ
4	1	2	FEC
5	0	4	ARQ
5	1	3	
5	2	2	FEC
6	0	5	
6	1	4	
⋮	⋮	⋮	

- Productivity-Bereich setzt eher auf die Fehlererkennung, weniger auf die Korrektur → ARQ-Protokoll (Automated Repeat Request) → es wird lieber eine neue Sendung angefordert

  ➔ Daten analysieren, Prozesse optimieren
- Broadcast, Telco → Broadcast hat keinen Rückkanal (wäre dieser vorhanden → wenn sich ein Endgerät melden würde, müsste an alle neu gesendet werden), im Telcobereich müssen die Verbindungen während des Gesprächs laufen, neu senden macht keinen Sinn

  ➔ FEC (Forward Error Correction)
- Bsp. für $d_{\min} = 3$ → Fehlerwörter sind korrekten Wörtern zugeordnet (würde ein bestimmtes Fehlerwort auftreten, würde es bei einer Korrektur in das entsprechend zugeordnete Wort geändert) → Schutzhülle für Codes

Zusammenhang

…zwischen der Codelänge $n = m + k$ und der Prüflänge m für $t = 1$

$$\frac{\text{Volumen aller Codewörter (gesamter n-dimensionaler Raum)}}{\text{Volumen eines Wortes}} \geq \text{Anzahl der Wörter}$$

$$\frac{2^n}{1+n} \geq 2^k \Rightarrow 2^m \geq n+1$$

k = Anzahl der Bits des korrekten Codes

m = Anzahl der Redundanzbits zur Fehlererkennung

Übersichtstabelle:

n	m	k	Redundanz
3	2	1	67 %
10	4	6	40 %
100	7	93	7 %
1000	10	990	1%
10000	14	99986	0,014 %

➔ Eine Verlängerung der Rahmenlänge führt zu einer Optimierung der Codierung (Redundanz durch Fehlererkennungsbits wird kleiner)

Mit einer Prüflänge m bei $t = 2$ ergibt sich entsprechend:

$$\frac{2^n}{1+n+\binom{n}{2}} \geq 2^k \Rightarrow 2^m \geq 1+n+\binom{n}{2}$$

Korrekturbits

- Werden zum Beispiel als Paritätsbits verwendet (z.B. 4-Bit Wert mit 3 Paritäten) → werden als Polynome interpretiert

  $1011 \Rightarrow 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$
- Codiertechnisch kann man mit den Koeffizienten wie in der Mathematik rechnen, wenn man sie entsprechend auffasst (Carry-Bit entfällt)
- Die einzelnen Bits sind linear unabhängig, also orthogonal zu einander

Beispiel zum Senden eines Wortes und Korrigieren eines Fehlers

- Die Erzeugung der Paritätsbits wird mit einer entsprechenden Matrix geregelt → in diesem Fall steht das Paritätsbit auf 0, wenn die Anzahl der Einsen im entsprechenden Vektor gerade ist → bei einer ungeraden Anzahl von Einsen im Vektor steht es entsprechend auf 1

Daten	1	0	1	1	P_1	P_2	P_3
	_	_		_	0		
	_		_	_		1	
		_	_	_			0
Senden	1	0	1	1	0	1	0
Empfangen	1	0	0	1	0	1	0
Prüfung	1	0	0	1			
	_	_		_	0		
	_		_	_		0	
		_	_	_			1
Syndromwort					0	1	1

Mit Hilfe des Syndromwortes und der zu Grunde gelegten Matrix kann man die fehlerhafte Stelle im Datenwort finden.

Syndromwort	Matrix der Paritäten-Erstellung				Paritätenmatrix		
0	_	_		_	_		
1	_		_	_		_	
1		_	_	_			_

In der Spalte, an denen Einsen im Syndromwort stehen und Matrixstellen existieren, liegt ein Fehler vor.
Spielt man den Ablauf mit 2 Fehlern durch, stellt man fest, dass man die Fehlerstelle nicht finden kann. Jenes lässt sich auch begründen: In einen 4-Bit System kann $d_{\min}$, also der geringste Abstand zwischen zwei korrekten Bits, maximal 3 sein. In diesem Fall kann aber (s. Tabelle) höchstens ein Fehler korrigiert werden. Der Versuch, 2 Fehler zu korrigieren, schlägt daher fehl.