Nachrichtentechnisches Praktikum

Videocodierung

Version 2.2
11.10.2017
Erstellt von: cand. el. Holger Meuel
Überarbeitet von: Bastian Farkas (2011),
Morten Diestelhorst (2018),
Dirk Ritzi (2018)

Institut für Nachrichtentechnik

Inhaltsverzeichnis

1. Einleitung: Motivation zur Videocodierung

Der vorliegende Versuch soll einen Einblick in das umfangreiche Thema der Videocodierung geben.

Warum aber wird überhaupt Videocodierung benötigt? Dazu sei folgende Überlegung angestellt: Wenn Fernsehen in gewohnter Auflösung erlebt werden soll, werden 625 Zeilen benötigt. Spaltenpixel im eigentlichen Sinne sind in der Bildröhre nicht definiert, da ein Elektronenstrahl ein Bild erzeugt, indem er die Bildschirmfläche zeilenweise abfährt und auf der Oberfläche aufgebrachte Leuchtstoffe zum Leuchten anregt. Der *CCIR 601 Studiostandard*[1] definiert in Anlehnung an den gängigen analogen Fernsehstandard für ein Digitalfernsehsignal eine Abtastfrequenz von 27 MHz bei einer Auflösung von 720×576. Das entspricht einem Bildseitenverhältnis von zirka $4:3$.

Für die Farbcodierung kann entweder das im PC-Bereich gängige RGB-Format verwendet werden, bei dem durch Angabe dreier Spektralstrahler für drei Primärvalenzen (darunter kann man sich *Grundfarben* vorgestellen, aus denen jede darstellbare Farbe innerhalb eines bestimmten Farbbereiches dargestellt werden kann) – etwa **R**ot, **G**rün, **B**lau (RGB) – beliebige Farben durch additive Farbmischung erzeugt werden.

Sowohl wegen historischer Gründe aus der Zeit der analogen Fernsehübertragung als auch wegen praktischer Erwägungen in Bezug auf die Datenreduktionsmöglichkeiten, auf die später genauer eingegangen wird, wurde jedoch die Farbkomponentendarstellung (YUV) gewählt, bei der Bildhelligkeit (Luminanz Y) und Farbinformation (Chrominanzkomponenten U und V) getrennt übertragen werden.[2] Eine vollständige Beschreibung des Bildinhalts inklusive der Farben liegt vor, wenn neben dem Luminanzsignal zwei Farbdifferenzsignale zur Verfügung gestellt werden, da sich daraus alle Farbinformationen berechnen lassen.

Beide Darstellungsarten lassen sich ineinander überführen, in vorliegendem Skript soll aufgrund des großen Anwendungsgebietes in der Fernsehübertragung jedoch bei der Variante mit getrennter Helligkeit und Farbe verblieben werden.

Wird wiederum in Anlehnung an das analoge Fernsehen eine Bildwiederholfrequenz von 50 Hertz ($50\,\frac{\text{Halbbilder}}{\text{s}}$) zugrunde gelegt und die oben erwähnte Abtastfrequenz von 27 MHz eingesetzt, ergibt sich für bewegte Farbbilder mit einer Quantisierung von 8 bit eine Datenrate von jeweils $27\,\text{MHz} \times 8\,\text{bit} \approx 216\,\frac{\text{Mbit}}{\text{s}}$ für die Luminanz und die beiden Farbdifferenzsignale, insgesamt fällt also eine Bruttodatenrate von $3 \times 216\,\frac{\text{Mbit}}{\text{s}} = 648\,\frac{\text{Mbit}}{\text{s}}$ an.

Selbst wenn ein nach oben erwähntem Studiostandard digitalisiertes Signal – wie man es heute beim digitalen *Standard-Definition Television* (SDTV) in vielen Haushalten findet – mit der Auflösung von 720×576 eingesetzt wird, benötigte die Übertragung im uncodierten Zustand eine Datenrate von etwa $166\,\frac{\text{Mbit}}{\text{s}}$.[3] Eine solche Kapazität lässt sich bei weitem nicht in einem

[1] Comité Consultatif International des Radiocommunications

[2] Das Farbsignal sollte in dem bereits vorgegebenen Frequenzband des etablierten Schwarz-Weiß-Fernsehsignals untergebracht werden. Zudem sollte das Farbfernsehen empfängerseitg abwärtskompatibel sein, d. h., dass auch Schwarzweiß-Empfänger weiterhin in der Lage sein sollten, das Fernsehsignal darzustellen, wenngleich natürlich auch nur unbunt.

[3] Dieser Zahl liegt eine Quantisierung von 8 bit und eine Bildwiederholfrequenz von $25\,\frac{\text{Vollbildern}}{\text{s}}$ zugrunde, welche rein rechnerisch äquivalent zu den oben erwähnten $50\,\frac{\text{Halbbildern}}{\text{s}}$ wäre. Zudem ist von einer Unterabtastung der Chrominanz im Vergleich zur Luminanz von 4:2:2 ausgegangen worden. Nähere Erläuterungen zur Herleitung entnehme man Kapitel 3.

herkömmlichen Fernsehkanal (der Bandbreite 5 MHz) unterbringen.[4] [7]

Durch Videocodierung soll die Datenmenge reduziert werden, die zur Darstellung von Bewegtbildern benötigt wird. Wie gezeigt wurde, ist sie zwingend notwendig, wenn die riesigen Datenmengen digitaler Bildsysteme ökonomisch und ressourcenschonend übertragen werden sollen. Oberste Priorität soll dabei sein, dass die wahrnehmbare Qualität idealerweise unverändert bleiben soll. Dazu bieten sich verschiedene Techniken an, die im aktuellen Versuch vorgestellt werden sollen.

Zielsetzung ist es, grundlegende theoretische Kenntnisse in der Quellencodierung zu vermitteln bzw. zu wiederholen und zu verfestigen.

Im Versuch sollen diese anhand anschaulicher Beispiele illustriert werden, wobei der bzw. die Studierende selber den Einfluss einzelner Komponenten einer Encoder-Decoder-Kette erleben soll.

Vor dem Versuch sind **Hausaufgaben** anzufertigen. Aufgabenstellungen sind in den Kapiteln im Abschnitt II zu finden. Die Ergebnisse werden im Rahmen eines Kolloquiums vor der eigentlichen Versuchsdurchführung besprochen. In der je nach Vorbereitungsstand der Versuchsteilnehmerinnen und -teilnehmer etwa dreiviertelstündigen Besprechung werden von der Übungsleiterin bzw. dem Übungsleiter theoretische Kenntnisse aus vorliegendem Skript abgefragt. Sollte die Betreuerin bzw. der Betreuer des Versuches den Eindruck gewinnen, dass sich einzelne Teilnehmer nicht oder nur unzureichend mit der Materie auseinandergesetzt haben, kann sie oder er die erfolgreiche Teilnahme von der Ausarbeitung zusätzlicher schriftlicher Hausaufgaben abhängig machen, die in einem angemessenen Zeitraum nach dem Versuchstermin einzureichen sind, oder die schlecht vorbereiteten Teilnehmer erneut zu einem späteren Termin einladen. Sollte erneut keine genügende Beschäftigung mit dem Thema *Videocodierung* zu bemerken sein, muss das Praktikum als *nicht bestanden* gewertet werden.

Im Anschluß an die theoretische Einführung sind die Aufgabenstellungen für die Versuchsteile im Abschnitt 14 zu finden, die während der Versuchszeit durchzuführen sind. Die Ergebnisse sollen in den vorgefertigten Protokollen im Kapitel 15 vermerkt werden und gemeinsam mit der Betreuerin oder dem Betreuer interpretiert werden.

[4] Die Bandbreite eines idealen (rauschfreien) Kanals in $\frac{\text{Mbit}}{\text{s}}$ berechnet sich zu nach dem Shannon-Hartley-Gesetz zu $C_N = 2 \cdot B$, wobei B die Bandbreite des Kanals in Hz ist.

2. Was in diesem Versuch *nicht* behandelt wird

Vorliegender Versuch befaßt sich ausschließlich mit dem Thema Quellencodierung, also dem Problem der senderseitigen Datenreduktion des Rohmaterials und empfängerseitigen Rückgewinnung und Adaption an das jeweilige Darstellungsmedium. Beispielsweise würde die Übertragung von hochaufgelöstem Videomaterial an einen mobilen Empfänger mit einem nur wenige Zoll messenden Display wenig Sinn ergeben, ebensowenig allerdings würde es einem Zuschauer Vergnügen bereiten, auf einem 40" Display Material geringster Auflösung – etwa QCIF[1], wie es für den Mobilempfang Verwendung findet – zu betrachten.

Explizit ***nicht*** betrachtet werden in diesem Versuch Themen der Kanalcodierung, jenes Teilgebiet der Nachrichtentechnik, welches sich mit dem Schutz eines Videosignals vor Übertragungsfehlern jeglicher Art befaßt, die auf einem realen Übertragungskanal zwangsläufig auftreten, seien es Störungen in einem Funkkanal, eines Satellitenkanals oder Kratzer und Verdreckung einer DVD.

All jene Themen werden in einem extra Versuch behandelt, der sich ausschließlich mit Kanalcodierung befaßt, insofern kann dieses ebenfalls wichtige Thema an dieser Stelle getrost vernachlässigt werden.

In der Abbildung 2.1 aus [5] ist ein komplettes, auf die wesentlichen Komponenten beschränktes, digitales Übertragungssystem dargestellt. Im Versuch wird sich mit den dick eingerahmten Blöcken am Anfang und am Ende befaßt werden. Dabei wird davon ausgegangen, dass das zu kodierende Bildmaterial bereits in digitaler Form vorliegt und wir unser Hauptaugenmerk auf den Block der Quellencodierung richten können.

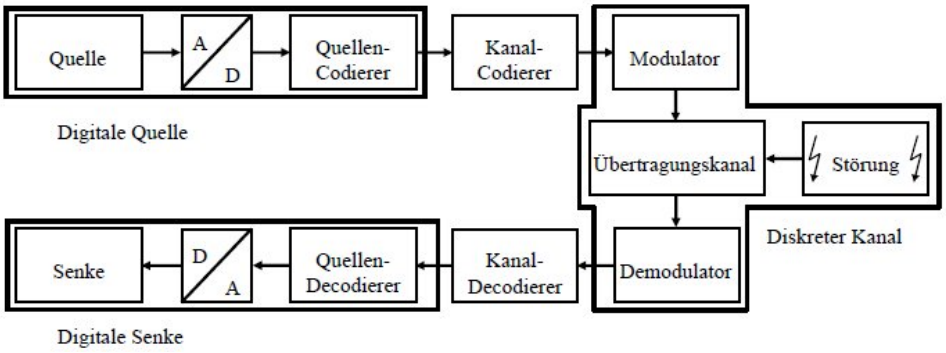


Abbildung 2.1.: Kanalmodell

[1] *Quarter Commom Intermediate Format*, also eine Auflösung von $144 \times 176\, Bildpunkten^2$ und $25\, \frac{Bilder}{s}$

Teil I.

Theoretische Grundlagen der Videocodierung

3. Einfache Verfahren der Bilddatenreduktion

Damit auch für den Versuchsteilnehmer ohne Vorwissen in der Videocodierung der Versuch verständlich und lehrreich ist, müssen zu Beginn einige Grundlagen der Theorie der Videocodierung eingeführt werden.

Wie bereits besprochen, sollen in diesem Versuch Techniken und Anwendungsgebiete der Videocodierung näher gebracht werden. Die Kanalcodierung, die sich mit dem Schutz der Übertragung vor Fehlern beschäftigt, wird dabei außen vor gelassen, da sich ein gesonderter Versuch im Rahmen der Versuchsreihe des *Nachrichtentechnischen Praktikums* dem Thema widmet.

Schon bei *Standard-Definition TV* (SDTV) ist die Datenrate zur Übertragung der Rohdatenmenge eines Videostreams mit $648\,\frac{\text{Mbit}}{\text{s}}$ gigantisch und ohne Bilddatenreduktion praktisch nicht durchführbar. [7] beschreibt geeignete Maßnahmen zur Datenreduktion, die nachfolgend zusammengefaßt werden sollen.

Eine triviale Methode der Datenreduktion besteht zunächst darin, statt der progressiv abgetasteten 50 Vollbilder pro Sekunde nur 50 *Halb*bilder (rechnerisch also entsprechend 25 Vollbildern) wie beim analogen Fernsehen im sogenannten Interlace-Verfahren, zu verwenden. Dadurch wird schon einmal die Hälfte der Datenrate, vulgo $324\,\frac{\text{Mbit}}{\text{s}}$, erreicht.

Wird zudem ausgenutzt, dass das menschliche Auge Details in Helligkeitsverläufen bis zu einem höheren Detaillierungsgrad (Ortsfrequenz) wahrnimmt als in der Chrominanzinformation, kann das Bild horizontal unterabgetastet werden – das bedeutet, dass wie in Abbildung 3.1 (angelehnt an [7]) dargestellt, zeilenweise doppelt so viele Luminanz wie Chrominanzwerte verwendet werden. Dieser Abtastmodus trägt die Bezeichnung 4:2:2. Dabei steht die erste Zahl für den Abtastratenfaktor der Luminanz und die beiden weiteren Ziffern für die Faktoren der horizontalen sowie vertikalen Farbdifferenzwerte.

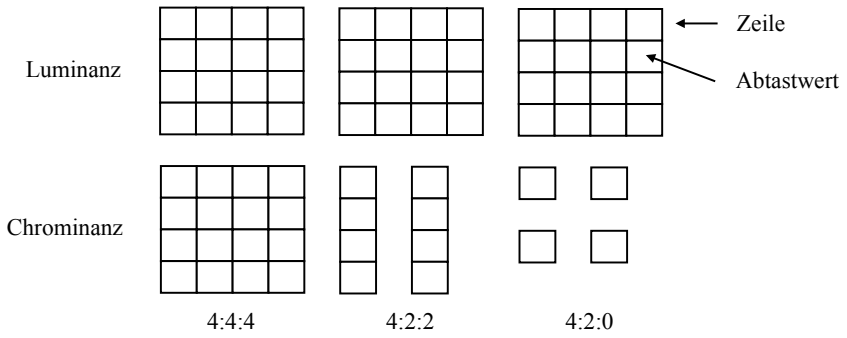


Abbildung 3.1.: Unterschiedliche Arten der Chrominanz-Abtastung

Da Austastlücke sowie Synchronisierimpuls nicht mehr benötigt werden, genügt ohne Qualitätsverlust eine Beschränkung auf den aktiven Bildbereich. Es wird eine Datenrate von $166\,\frac{\text{Mbit}}{\text{s}}$ erreicht, die – technisch gesehen nicht 100% korrekt – immer noch als *unkomprimiertes* Videomaterial bezeichnet wird.

Als letzten Schritt können die Farbsignale vertikal ebenfalls unterabgetastet werden. Dabei werden nurnoch in jeder zweiten Zeile Farbdifferenzwerte abgetastet. Dies entspricht der sogenannten 4:2:0-Abtastung. Hierdurch wird immerhin eine Datenratenreduktion um den Faktor

fünf erreicht, ohne dass der menschliche Betrachter davon etwas bemerken würde.

Wird eine weitergehende Datenreduktion angestrebt, müssen entweder weitere Irrelavanzreduktionsmaßnahmen ergriffen werden oder aber Maßnahmen, die über reine Irrelevanzreduktion hinausgehen, wodurch das Videomaterial allerdings sichtbar an Qualität einbüßt. Wenn allerdings die Anforderungen an die Auflösung – wie etwa im Falle des Mobilempfangs auf kleinen Displays – ohnehin nicht besonders hoch sind, genügt auch eine geringere Auflösung wie sie etwa das *Common Intermediate Format* (CIF) bereitstellt. Genauere Ausführungen dazu sowie Definitionen der Begriffe *Relevanz* und *Irrelevanz* folgen im Kapitel 4.2.

Die Tabelle 3.1 ist aus [7] entnommen und stellt die beschriebenen Schritte noch einmal übersichtlich zusammen.

Parameter	**Datenrate** **Luminanz**	**Chrominanz**	**Gesamt**
Progressive Bildabtastung, 50 $\frac{\text{Bilder}}{\text{s}}$, 625 Zeilen, Abtastfrequenz 27 MHz für Luminanz und Chrominanz	216	216 + 216	648
Zwischenzeilenverfahren, Abtastfrequenz 13,5 MHz	108	108 + 108	324
Unterabtastung der Chrominanz mit der Abtastfrequenz 6,75 MHz („4:2:2“)	108	54 + 54	216
Beschränkung auf den aktiven Bildbereich, $(720 \times 576 \cdot 8 \cdot (1 + 0,5 + 0,5) \cdot 25 \frac{\text{Bilder}}{\text{s}})$	83	41,5 + 41,5	166
Vertikale Unterabtastung der Chrominanz („4:2:0“)	83	20,7 + 20,7	125
Reduktion der Luminanz-Auflösung, [Common Intermediate Format (CIF)], 25 $\frac{\text{Bilder}}{\text{s}}$, 288 Zeilen, 352 Bildpunkte	20,7	5,2 + 5,2	31

Tabelle 3.1.: Triviale Möglichkeiten der Datenreduktion

Bisher wurden sowohl die Reduktion der örtlichen Auflösung (Pixel pro Bild) wie auch mit der zeitlichen Auflösung (Bildwiederholfrequenz) betrachtet. Als dritter Freiheitsgrad stünde noch die Farbtiefe (Bits pro Farbe) zur Verfügung, die jedoch unangetastet bleiben soll, da bei der Farbquantisierung kein Einsparpotential zur Verfügung steht. Eine weitere Quantisierung des Quellmaterials würde zwar einen weiteren Gewinn bringen, jedoch sichtbare Chrominanzfehler nach sich ziehen.

4. Grundlagen der Informationstheorie

Damit überhaupt sinnvoll mit Themen der Videocodierung umgegangen werden kann, werden ein paar Grundbegriffe der Informationstheorie, die im folgenden vorgestellt werden, benötigt. Diese Begrifflichkeiten sollte sich jeder, der im Bereich Nachrichtentechnik tätig sein möchte, dringend einprägen, da sie einem in jeder Teildisziplin wieder begegnen werden.

Dabei werden Begriffe wie *Nachricht* oder *Information*, die in der Umgangssprache täglich vorkommen, verwendet. Im fachsprachlichen Kontext allerdings bedarf es einer genauen Definition.

4.1. Information

Durch eine *Information* wird ganz allgemein der Kenntnisstand des Empfängers vergrößert.

4.2. Nachricht

Eine *Nachricht* ist eine Folge von (vorher vereinbarten) Zeichen, die zur Übermittlung von *Informationen* dient.

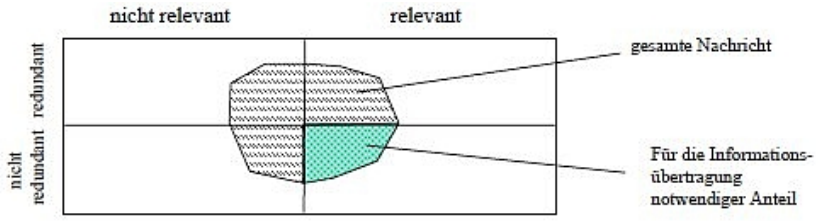


Abbildung 4.1.: Nachricht aus Sicht der Informationstheorie

Wie in der Grafik 4.1 aus [4] ersichtlich, bedarf es gemäß [7] einer Unterteilung von Nachrichten in einen interessierenden Teil (relevanter Teil) und einen nicht-interessierenden Teil (Irrelevanz) und zudem in redundante (überflüssige) und nicht-redundante Information. Interessant im Sinne der Informationstheorie ist lediglich der **nicht-redundante, relevante** Teil.

Das bedeutet, dass für die Videocodierung weder Bildanteile, die vom menschlichen Wahrnehmungsapparat ohnehin nicht verarbeitet werden – etwa farbliche Details hochaufgelöster Strukturen – interessant sind, noch müssen mehrfach enthaltene Informationen auch mehr als einmal übertragen werden. Eine einmalige Übertragung mit dem Hinweis, an welchen Stellen im Bildmaterial diese Information benötigt wird, genügt vollkommen. Statt also für eine schwarze Bildzeile 576 mal den Wert *schwarz* zu übertragen, ist es zweckdienlicher im Sinne der Datenreduktion, nur einmalig den Wert *schwarz* zu übertragen und als Zusatzinformation mitzusenden, dass dieser Wert für die ganze Zeile gelten soll.

4.3. Entscheidungsgehalt

Der *Entscheidungsgehalt* H_0 einer Quelle beschreibt die Anzahl der für die Auswahl eines *Zeichens* notwendigen Binärentscheidungen. Man kann sich diese Binärentscheidungen als Pfad in einem Entscheidungsbaum wie in Abbildung 4.2 vorstellen. Hier hat die Quelle vier Zeichen (Ziffern

1 bis 4). Treten alle Zeichen der Quelle mit der gleichen *Wahrscheinlichkeit* auf, so werden exakt zwei Entscheidungen zur eindeutigen Wahl eines Zeichens benötigt. Demnach beträgt der *Entscheidungsgehalt* $2\,\frac{\text{bit}}{\text{Zeichen}}$ und lässt sich folgendermaßen berechnen:

$$H_0 = \text{ld}(N) \tag{4.1}$$

Die (Pseudo-)Einheit lautet: $[H_0] = \frac{\text{bit}}{\text{Zeichen}}$. N ist dabei die Anzahl verschiedener Zeichen einer Quelle. *ld* steht für den *logarithmus dualis*, also den Logarithmus zur Basis zwei. Für binäre Systeme ist $\text{ld} := \log_2$ eine nützliche Schreibweise, die auch in der Literatur desöfteren Verwendung findet. *bit* steht hier nicht für die binäre Speicherinheit, sondern für die binäre *Entscheidung*.

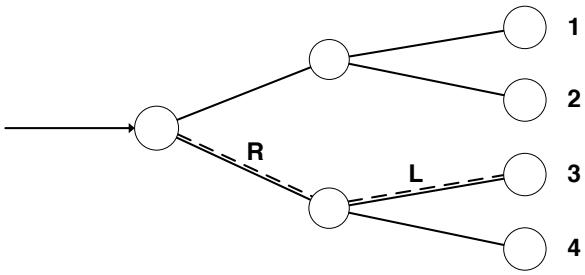


Abbildung 4.2.: Binärer Entscheidungsbaum

4.4. Wahrscheinlichkeit

Die *Wahrscheinlichkeit* $p(x_i)$ ist definiert als das Verhältnis der Anzahl des Auftretens eines Ereignisses (sogenannte *günstige* Ereignisse) zur Gesamtanzahl der Stichprobe.

4.5. Informationsgehalt

Der *Informationsgehalt* $I(x)$ gibt die Bedeutung eines *Ereignisses* in Abhängigkeit seiner *Wahrscheinlichkeit* $p(x_i),\ i = 1 \ldots N$ an. Hierbei wird angenommen, dass ein Ereignis $X = \{x_i\}$ – also das Eintreten eines bestimmten Symbols – mit der Wahrscheinlichkeit $p(X)$ eintritt und $x_1, ..., x_N$ Symbole aus einem N-elementigen Alphabet sind.

$$I(x_i) = \text{ld}(\frac{1}{p(x_i)}) \tag{4.2}$$

Als anschauliches Beispiel sei hier der Wetterbericht genannt. Im Sommer hat das Ereignis „Sonne“ eine sehr hohe Wahrscheinlichkeit. Der Informationsgehalt der Nachricht „Morgen scheint die Sonne“ ist also relativ gering.

4.6. Entropie

Die Entropie $H(X)$ einer Quelle gibt die *Menge an Zufall* – also den *mittleren „Überraschungsgehalt“* – an, der in einem System steckt. Sie ist der *mittlere Informationsgehalt* und errechnet sich zu:

$$H(X) = \langle I(x_i) \rangle = \sum_{i=1}^{N} p(x_i) \cdot \text{ld}(\frac{1}{p(x_i)}) \tag{4.3}$$

Ein sicheres Ereignis $(p = 1)$ hat ebenso wie ein unmögliches Ereignis $(p = 0)$ die Entropie 0, weil in beiden kein Zufall steckt.

Angeknüpft an Abschnitt 4.5 ein kleines Rechenbeispiel:

Seien die möglichen Ereignisse eines Wetterberichts: „sonnig“, „bewölkt“, „regnerisch“. Weiterhin wird angenommen, dass alle Ereignisse mit der gleichen Wahrscheinlichkeit eintreten können. Dann berechnet sich die Entropie des „Wetterberichtes“ zu:

$$H_{Wetter} = \sum_{i=1}^{3} 0.33 \cdot \mathrm{ld}(\frac{1}{0.33}) = 0.33 \cdot 1.58 + 0.33 \cdot 1.58 + 0.33 \cdot 1.58 = 1.58$$

Die *Menge an Zufall* ist hier maximal, da nicht gesagt werden kann wie das Wetter wird.

Es wird nun angenommen, dass „sonnig“ eine Wahrscheinlichkeit von 80% hat und die beiden anderen Ereignisse jeweils 10%. Dann ergibt sich die Entropie zu:

$$H_{Wetter} = 0.8 \cdot 0.32 + 0.1 \cdot 3.32 + 0.1 \cdot 3.32 = 0.92$$

Hier ist die *Entropie* sehr gering, da ein Ereigniss mit überdurchschnittlich hoher Wahrscheinlichkeit eintritt.

Multipliziert man die Entropie $H(X)$ mit der Anzahl der zu codierenden Symbole eines Wortes, so kann das Ergebnis als die Größe einer **optimalen** Codierung des Wortes in Bit verstanden werden.

5. Statistische Modelle

Unter dem Begriff *Statistische Modelle* sind verschiedene Codierungsarten zusammengefaßt. Gemein ist allen, dass statistische Eigenschaften zur Effizienzsteigerung der Codierung ausgenutzt werden, beispielsweise die Auftretenswahrscheinlichkeit eines Symbols bei der Entropiecodierung, mehrere hintereinanderfolgende gleiche Symbole bei der Lauflängencodierung oder aber gleiche Strukturen im Bildmaterial bei der Prädiktionscodierung.

5.1. Entropiecodierung

Als *Entropiecodierung* bezeichnet man Codierverfahren, mit denen Wörter so codiert werden, dass die Zeichen in ihrer codierten Repräsentation im Durchschnitt möglichst kurz werden. Dabei wird versucht, möglichst nahe an die theoretische untere Grenze – die Entropie – heranzukommen. Die Codierung erfolgt unter Ausnutzung statistischer Modelle und ist verlustfrei.

Für eine Entropiecodierung ist es zweckdienlich, für die Informationssymbole verschieden lange Codewörter in Abhängigkeit ihrer Auftretenswahrscheinlichkeit zu verwenden. Je häufiger ein Wort auftritt, desto kürzer sollte sinnvollerweise das es repräsentierende Codewort sein, da es schließlich **im Mittel** sehr häufig übertragen werden wird. Prominenter Vertreter dieser sogenannten *Variable Length Codes* (VLC) ist die *Huffman-Codierung*

Eine effizientere Entropiecodierung ist die *arithmetische Codierung*, die nicht wie die Huffman-Codierung auf eine ganzzahlige Anzahl Bits zur Repräsentation eines Zeichens angewiesen ist. Deshalb ist sie effizienter als die Huffman-Codierung und findet in modernen Systemen wie beispielsweise dem H.264 / AVC-Standard (vergleiche Kapitel 9.2) Verwendung.

Beide genannten Entropiecodierungen besitzen die *Präfixeigenschaft*, welche besagt, dass kein Codewort der Anfang eines anderen Codeworts sein darf. Codes dieser Klasse haben den immanenten Vorteil, dass keine Zusatzinformation übertragen werden muss, die dem Empfänger den Beginn eines neuen Symbols mitteilt.

Der einfachste VLC ist die sogenannte *unäre Codierung*, bei der einer zu codierenden Zahl n als Codewort n mal die 0 und eine abschließende 1 zugeordnet wird. Die 3 hätte beispielsweise das Codewort „0001“. Da die Länge L eines jeden Codeworts ganzzahlig ist und nicht kleiner als eins sein kann, die Entropie eines Wortes aber im Allgemeinen nicht ganzzahlig ist und auch kleiner sein kann als eins, kann im Allgemeinen mit einem VLC die Vorgabe, ein Wort im Sinne der Entropie optimal zu codieren, nicht erfüllt werden. Diese Schwäche wird häufig dadurch zu umgehen versucht, dass Symbole, die sehr häufig hintereinander auftreten, zusammengefaßt werden, indem die Häufigkeit des Auftretens codiert wird und nicht mehrmals hintereinander dasselbe Symbol. Man spricht dann von einer *Lauflängencodierung*.

Nach dem Shannon'schen Codierungstheorem [4, 8] lässt sich für jede Quelle eine Binärcodierung finden, sodass gilt:

$$H(X) \leq \overline{L} < H(X) + 1, \tag{5.1}$$

wobei $\overline{L}$ die mittlere Länge aller Codewörter der Quelle ist.

Der Huffman-Code wird im folgenden Unterkapitel 5.1.1 besprochen, die arithmetische Codierung soll hier aus Platzgründen nicht weiter thematisiert werden. Sie ist komplizierter, jedoch wegen ihrer höheren Effizienz diejenige Entropiecodierung, die in aktuellen Systemen wie H.264 / AVC / MPEG-4 part10 eingesetzt wird.

5.1.1. Huffman-Codes

Ein häufig verwendeter Vertreter der VLCs ist die Huffman-Codierung, die hier kurz anhand eines Beispiels erläutert wird.

Gegeben sei ein Alphabet $A = \{a, b, c, d, e\}$ und eine Auftrittswahrscheinlichkeit für jedes Symbol $x_i \in A$.

$$\begin{aligned} p(\text{a}) &= \frac{1}{10} \\ p(\text{b}) &= \frac{1}{10} \\ p(\text{c}) &= \frac{2}{10} \\ p(\text{d}) &= \frac{3}{10} \\ p(\text{e}) &= \frac{3}{10} \end{aligned}$$

Für die Bildung des Codes wird ein Baum wie in Abbildung 5.1 aufgestellt.

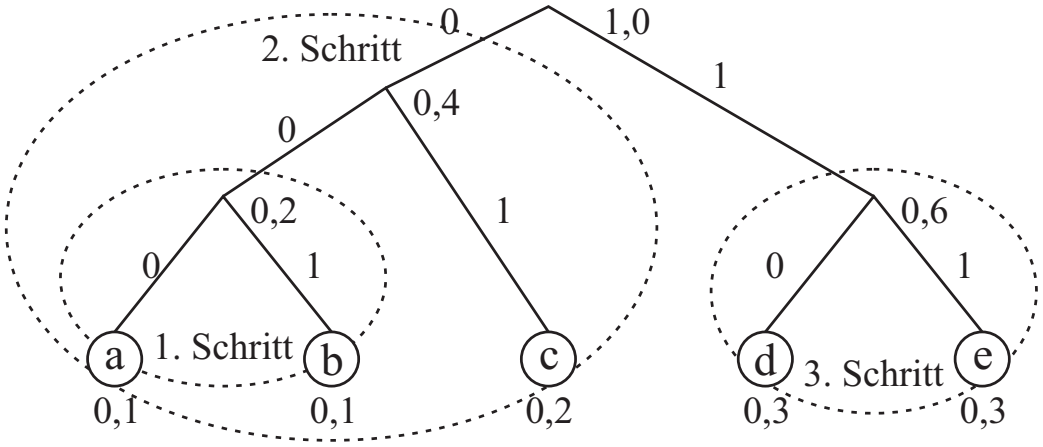


Abbildung 5.1.: Bildung eines Huffman-Codes

Dafür wird zunächst jedem Symbol ein Knoten zugeordnet, der als Wert die Auftrittswahrscheinlichkeit des entsprechenden Symbols zugewiesen bekommt. Anschließend werden sukzessive die beiden Knoten mit den geringsten Werten (Wahrscheinlichkeiten) durch einen neuen Knoten verbunden, der die Summe der beiden einzelnen Knotenwerte zugewiesen bekommt. Dieses wird so lange wiederholt, bis alle ursprünglichen und neu entstandenen Knoten zu einem Baum verbunden sind. Die codierte Repräsentation eines Symbols lässt sich nun finden, indem man in der Wurzel des Baumes startet und bis zu dem Blatt herabsteigt, das zu dem gewünschten Symbol gehört. Ein Schritt entspricht dabei dem Hinzufügen einer 0 oder einer 1 im Codewort. Dabei kann prinzipiell beliebig entschieden werden, welchem Ast man welches Bit zuweist. Es ist aber sinnvoll, hier nach festen Regeln vorzugehen. Im Beispiel wurde so vorgegangen, dass ein Schritt in Richtung des Knotens mit dem kleineren Wert einer 0 entspricht. Bei gleichen

Knotenwerten wurde die 0 dem Knoten zugeordnet, der „weiter links“ liegt. Somit ergibt sich folgende Code-Tabelle.

a	000
b	001
c	01
d	10
e	11

Da jedes Blatt durch einen eindeutigen Pfad zu erreichen ist, ohne dass ein anderes Codewort passiert wird – also kein Codewort *Präfix* eines anderen ist – sind diese Repräsentationen eindeutig decodierbar.

Codiert man zum Beispiel das Wort

abccdddeee,

ergibt sich als Codewort

0000010101101010111111.

Daher ist hier $\overline{L} = \frac{22}{10}\,\text{bit} = 2,2\,\text{bit}$, d. h. ein Symbol der Quelle benötigt *im Mittel* in seiner codierten Repräsentation $2,2\,\text{bit}$. Die übliche binäre Codierung würde bei fünf Symbolen $\text{ld}(8)\,\text{bit} = 3\,\text{bit}$ benötigen. Die Entropie des Wortes ergibt sich zu

$$\begin{aligned} H(X) &= \left(\frac{1}{10} \cdot \text{ld}(10) + \frac{1}{10} \cdot \text{ld}(10) + \frac{2}{10} \cdot \text{ld}(\frac{10}{2}) + \frac{3}{10} \cdot \text{ld}(\frac{10}{3}) + \frac{3}{10} \cdot \text{ld}(\frac{10}{3}) \right) \text{ bit} \\ &= 2,17\,\text{bit} \end{aligned}$$

In der Tat ist damit hier gezeigt, dass der Huffman-Code das Shannon'sche Codierungstheorem (Gleichung 5.1) erfüllt.

Deshalb spricht man bei Huffman-Codes häufig von einer *optimalen Entropiecodierung*. Huffman selbst nannte seine Codes *minimum redundancy codes*.

5.2. Korrelation zwischen Symbolen zur Senkung der mittleren Codewortlänge

Wortlängen, die im Mittel kürzer sind als die Entropie lassen sich nur erreichen, wenn man die Korrelation zwischen Worten ausnutzt. Dabei wird die *Beziehung zwischen* Einzelsymbolen herangezogen: Am Beispiel der deutschen Sprache etwa sieht man, dass der Buchstabe *e* zwar sehr häufig auftaucht, dass einem *e* jedoch ein weiteres *e* folgt, ist relativ unwahrscheinlich. Anders herum kann man beim Empfang des Symbols *q* mit hoher Wahrscheinlichkeit voraussagen, dass das nächste Symbol ein *u* sein wird.

Der Preis, mit dem man sich diese zusätzliche Einsparung an mittlerer Codewortlänge erkauft, ist, dass der Codierungsaufwand unverhältnismäßig stark ansteigt, da die Zahl der möglichen Zeichen exponentiell wächst.

Zudem können andere Probleme wie beispielsweise *katastrophale Fehlerfortpflanzung* auftreten. Dieses Phänomen besagt, dass ein einmaliger Decodierfehler nicht wieder korrigiert werden kann. Genauere Ausführungen können [4] entnommen werden und werden in dem Versuch zur Kanalcodierung im Rahmen des *Nachrichtentechnischen Praktikums* vertieft.

6. Prädiktionscodierung

Bisher wurde durch geschickte Codewortzuweisung eine *verlustlose* Kompression um zirka den Faktor zwei ermöglicht. Höhere Codierungsgewinne sind nur noch verlustbehaftet möglich, was erst einmal nicht sehr verlockend klingt.

Bei näherer Analyse der menschlichen Sinne, auf welche schließlich alle Systeme optimiert werden sollten, stellt sich dann jedoch heraus, dass das Auflösungsvermögen ohnehin in diversen Aspekten begrenzt ist. Information, die über die Auflösungsfähigkeit der akustischen und optischen Sinnesorgane hinausgeht, wird ohnehin nicht bemerkt. Folglich kann man sie auch von vornherein entfernen und dadurch größere Gewinne als bisher möglich erzielen.

Die *Prädiktion* entpuppt sich bei näherer Betrachtung als hervorragende Möglichkeit der Datenreduktion von Bilddaten. Dabei muss zwischen örtlicher und zeitlicher Prädiktion unterschieden werden.

Das Wort *Prädiktion* stammt aus dem Lateinischen und bedeutet *Vorhersage*. In der Bild- und Videocodierung spricht man von Prädiktion, wenn Bildteile aus benachbarten Pixeln oder aus vorher decodierten Bildern der gleichen Videosequenz abgeschätzt werden. Bei einem gut prädizierten Bild müssen nur noch geringfügige Abweichungen des Originalbildes vom prädizierten in einem Differenzbild explizit abgespeichert werden. Die Wahrscheinlichkeit von kleinen Werten im Differenzbild ist also wesentlich größer als die von großen Werten, sodass sie mit den Mitteln aus Kapitel 5.1 effizienter gespeichert werden können.

6.1. Örtliche Prädiktion

Die *örtliche Prädiktion* nutzt aus, dass natürliches Bildmaterial viel Homogenität in dem Sinne enthält, dass benachbarte Pixel häufig zu einem dargestellten Objekt gehören und ähnliche oder gleiche Helligkeit besitzen, da Objekte häufig eine regelmäßige Oberflächenbeschaffenheit aufweisen. [6]

2	2	2	2	2	3	3	2
2	2	2	0	2	3	3	2
2	2	0	0	0	2	2	2
1	1	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

1	0	0	0	0	1	0	-1
1	0	0	-2	2	1	0	-1
1	0	-2	0	0	2	0	0
0	0	0	-1	1	0	0	0
0	0	-1	1	-1	1	0	0
0	0	0	0	0	0	0	0

Abbildung 6.1.: Beispiel für eine einfache örtliche Prädiktion: links das Originalbild, rechts das Differenzbild nach der Prädiktion

Ein einfaches Beispiel für eine Prädiktion ist, wenn man statt des Grauwertes eines jeden Pixels nur die Differenz seines Grauwertes zu dem seines linken Nachbarn speichert. Außerhalb des Bildes könnte man einen Standardwert annehmen. Abbildung 6.1 zeigt ein mit 2 bit quantisiertes Bild und sein prädiziertes Pendant. Als Standardwert wurde hier eine 1 verwendet.

Die Entropie des Bildes ergibt sich zu

$$\begin{aligned} H_B &= \frac{7}{48} \cdot \mathrm{ld}(\frac{48}{7}) + \frac{21}{48} \cdot \mathrm{ld}(\frac{48}{21}) + \frac{16}{48} \cdot \mathrm{ld}(\frac{48}{16}) + \frac{4}{48} \cdot \mathrm{ld}(\frac{48}{4}) \\ &= 1,75, \end{aligned}$$

die des Differenzbildes zu

$$\begin{aligned} H_F &= \frac{31}{48} \cdot \mathrm{ld}(\frac{48}{31}) + \frac{5}{48} \cdot \mathrm{ld}(\frac{48}{5}) + \frac{8}{48} \cdot \mathrm{ld}(\frac{48}{8}) + \frac{2}{48} \cdot \mathrm{ld}(\frac{48}{2}) + \frac{2}{48} \cdot \mathrm{ld}(\frac{48}{2}) \\ &= 1,56. \end{aligned}$$

Statt nur den linken Nachbarn zu verwenden, kann man diese Art der Prädiktion weiter verfeinern, indem man alle benachbarten Pixel verwendet, die zu dem Zeitpunkt, zu dem das aktuelle Pixel decodiert wird, bereits bekannt sind. Da Bilder in der Regel zeilenweise von links-oben nach rechts-unten gespeichert werden, sind das die Pixel, die in Abbildung 6.2 sind.

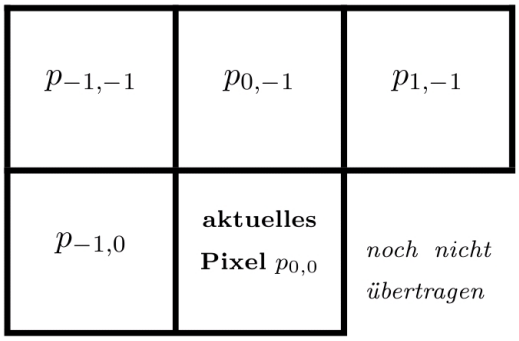


Abbildung 6.2.: Pixel, die für die örtliche Prädiktion von Bedeutung sind

$p_{0,0}$ ist dabei das aktuelle Pixel. Durch eine gewichtete Aufsummierung mit der Prädiktionsmatrix der umliegenden Grauwerte dargestellt entsteht der Prädiktor

$$\tilde{p}_{0,0} = d \cdot p_{-1,0} + b \cdot p_{0,-1} + a \cdot p_{1,-1} + c \cdot p_{-1,-1}. \tag{6.1}$$

Durch Optimierung der Parameter a, b, c und d ließe sich ein guter Prädiktor finden.

In der Videocodierung spricht man bei dieser Art der Prädiktion, bei der nur *Pixel des aktuellen Bildes* verwendet werden, von **Intra-Prädiktion**. Mit einer Intra-Prädiktion können Bilder auf etwa die Hälfte ihrer Datenmenge reduziert werden.

In der Videocodierung stehen für die Prädiktion weitere Daten aus *vorher* decodierten Bildern zur Verfügung. Werden diese genutzt, spricht man von **Inter-Prädiktion**. Die Abhängigkeiten zwischen Teilen aufeinander folgender Bilder (zeitlich) sind wesentlich größer als die zwischen benachbarten Pixeln (örtlich). Da deshalb der *zeitlichen Prädiktion* eine Schlüsselrolle bei der Videocodierung zukommt, ist ihr das komplette folgende Unterkapitel 6.2.1 gewidmet.

6.2. Zeitliche Prädiktion

Zeitliche Prädiktion funktioniert ähnlich wie örtliche Prädiktion, allerdings wird nicht die Information aus benachbarten Pixeln zur Vorhersage verwendet, sondern diejenige, die in anderen

Frames der Sequenz enthalten ist, weshalb diese Codierungsart als **Interframe-Prädiktion** bezeichnet wird.

Eine Verfeinerung der Prädiktionsqualität kann durch Kombination mit einer Bewegungsschätzung in Verbindung mit einer Bewegungskompensation erreicht werden. In jedem Falle ist es bei der zeitlichen Prädiktion notwenig, den Schätzfehler $r(n)$ als Restsignal sowie im Falle der verbundenen Bewegungsschätzung zusätzlich die ermittelten Bewegungsvektoren zu übertragen.

6.2.1. Zeitliche DPCM

Um diesem systematischen Fehler vorzubeugen, verwendet man eine Struktur, die *Differential Pulse Code Modulation* (DPCM) genannt wird. Sie besagt im Grunde nichts weiter, als dass wie in Abbildung 6.3 aus [7] dem Prädiktor das quantisierte Signal zugeführt wird und das prädizierte Signal wiederum per Rückkopplung vom Originalsignal abgezogen wird und erneut der Quantisierung zugeführt wird. Rechts am Ausgang liegt das *Rest-* oder *Fehlersignal* zur Übertragung an.

Nun benutzen Sender und Empfänger die *gleichen Werte* für die Prädiktion. Dadurch wird, der Darstellung in [6] folgend, der Quantisierungsfehler $d[e(m,n), v(m,n)]$ gleich dem Codierungsfehler $d[x(m,n), y(m,n)]$, wodurch der Effekt des „Auseinanderlaufens“ nach [6] vermieden wird:

$$\begin{aligned} r(m,n) &= e(m,n) - v(m,n) \\ &= [x(m,n) - \hat{x}(m,n)] - [y(m,n) - \hat{x}(m,n)] \\ &= x(m,n) - y(m,n). \end{aligned} \tag{6.2}$$

Dieses Restsignal muss neben den Prädiktionsparametern über den Kanal übertragen werden.

Das führt uns letztlich zur gesamten Encoder-Decoder-Kette wie in Abbildung 6.4 dargestellt (ebenfalls aus [7] entnommen): Hier ist nun final der Encoder aus Abbildung 6.3 in das Übertragungssytem eingesetzt. Es wird deutlich, dass Encoder und Decoder das gleiche Eingangssignal erhalten und dementsprechend der Decoder auch ein korrekt vorhergesagtes Bild liefert.

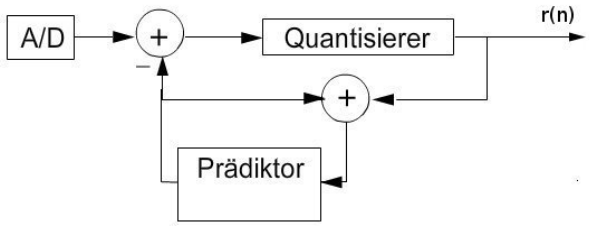


Abbildung 6.3.: (zeitlicher) DPCM-Encoder (korrekt)

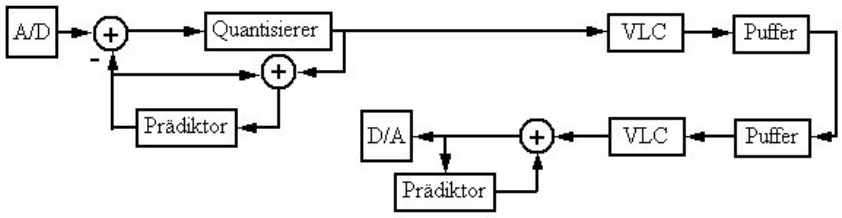


Abbildung 6.4.: DPCM-System zur Prädiktion

Zusammenfassend kann festgestellt werden, dass der große Vorteil des *DPCM*-Verfahrens in der *verlustlosen* Kompression liegt, jedoch sind die Codiergewinne aktueller Systeme mit Datenraten von $0,3 - 0,5 \frac{\text{bit}}{\text{Pixel}}$ im Vergleich zu den Transformationscodierungen geringer. [6]

7. Transformationscodierung

Werden die Spektren von zweidimensionalen Bildsignalen betrachtet, kann festgestellt werden, dass die Signalleistung bei natürlichen Bildern üblicherweise ihren Hauptanteil bei den niedrigen Frequenzen hat. Wird also ein Bild einer *diskreten Fourier-Transformation* (DFT)[1] unterzogen, erscheinen mit hoher Wahrscheinlichkeit nur bei den niedrigen Frequenzen große Koeffizienten.

Tests mit Probanden haben ergeben, dass das menschliche visuelle System weniger empfindlich auf hochfrequente Bildanteile – also kleine und filigrane Strukturen, wie sie häufig bei Rasenflächen, in Haaren, bei Bäumen mit einzelnen Blättern etc. zu finden sind – reagiert als auf niederfrequente wie die grobe Struktur des Bildes oder gar dem Gleichwert, der die Helligkeit eines Pixeln als niederfrequentester Koeffizient repräsentiert. Das bedeutet, dass Störungen, die durch eine stärkere Quantisierung bei den hochfrequenten Koeffizienten entstehen, einem Betrachter weniger auffallen als welche, die durch eine gröbere Abstufung bei den *niederfrequenten* Koeffizienten entstehen.

Abbildung 7.1 (aus [6]) demonstriert den Einfluss von fehlerhaften niederfrequenten Koeffizienten, die als deutlich sichtbare Helligkeitstörungen dem Betrachter ins Auge stechen, während möglicherweise vorhandene hochfrequente Störungen etwa im Bereich der Haare oder des Hutes wenig bis gar nicht auffallen.

Abbildung 7.1.: Auswirkung von Helligkeitsfehlern in Makroblöcken

Man könnte also die diskrete Fourier-Transformierte des Bildes noch platzsparender speichern, indem man ihre Koeffizienten unterschiedlich stark quantisiert. Das führt zwar zu einem Qualitätsverlust, der aber weniger sichtbar ist als der, der entstehen würde, wenn man das gesamte Bild stärker quantisieren würde.

Die Bildcodierung nach JPEG[2] zum Beispiel macht sich diese Eigenschaften zunutze. Allerdings wird als Transformation nicht die DFT verwendet, sondern die *diskrete Kosinus Transformation* (DCT). Sie hat reelle Basisfunktionen und ist deshalb besser zu implementieren. Außerdem wird nicht das gesamte Bild der DCT unterzogen, sondern das Material wird zunächst in

[1] Die *DFT* ist eine Transformation, die zu einem Ursprungsbild das dazugehörige Spektrum liefert. Das Spektrum repräsentiert die Verteilung der Frequenzanteile. Für weiterführende Erklärungen sei auf entsprechende Fachliteratur oder Vertiefungsvorlesungen wie *Digitale Signalverarbeitung*, *Signalverarbeitung I* oder *Grundlagen der Bildverarbeitung* verwiesen.

[2] *JPEG* steht für die *Joint Photographics Experts Group*, welche sich hauptsächlich mit der Standardisierung von Standbildern beschäftigt.

beispielsweise 8×8 große Blöcke zerlegt, die dann einzeln transformiert werden. Diese Blockgröße hat sich als sinnvoll herausgestellt, weil sie einerseits groß genug ist, um die statistischen Eigenschaften des Bildspektrums nutzen zu können, andererseits aber auch klein genug, damit sich Effekte, die durch lokale Eigenschaften des Bildes hervorgerufen werden, nicht durch größere Teile des Bildes fortsetzen.

Die 8×8-DCT des Blocks $b(x, y)$ ist durch die Formel 7.1 definiert. Abbildung 7.2 zeigt ihre zweidimensionalen Basisfunktionen.

$$\begin{aligned} DCT(f_x, f_y) &= \frac{1}{4} \cdot K(f_x) \cdot K(f_y) \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} b(x, y) \\ &\quad \cdot cos((2x+1) \cdot f_x \cdot \frac{\pi}{16}) \\ &\quad \cdot cos((2y+1) \cdot f_y \cdot \frac{\pi}{16}) \\ K(f) &= \begin{cases} \frac{1}{\sqrt{2}} & \text{für } f = 0 \\ 1 & \text{sonst} \end{cases} \end{aligned} \tag{7.1}$$

Ihre Inverse berechnet sich nach Formel 7.2 zu:

$$\begin{aligned} b(x, y) &= \frac{1}{4} \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} K(f_x) \cdot K(f_y) \cdot DCT(f_x, f_y) \\ &\quad \cdot cos((2x+1) \cdot f_x \cdot \frac{\pi}{16}) \\ &\quad \cdot cos((2y+1) \cdot f_y \cdot \frac{\pi}{16}) \\ K(f) &= \begin{cases} \frac{1}{\sqrt{2}} & \text{für } f = 0 \\ 1 & \text{sonst} \end{cases} \end{aligned} \tag{7.2}$$

f_x ist die Ortsfrequenz in x-Richtung, f_y entsprechend in y-Richtung. $K(f)$ dient wie der Vorfaktor $\frac{1}{4}$ lediglich zur Normierung, damit das Ergebnis der DCT in einen definierten Wertebereich (in der Regel 11 bit) abgebildet werden kann.

Die Quantisierung geschieht durch punktweise Division mit einer Matrix. Formel 7.3 zeigt ein Beispiel für eine solche Matrix, die durch umfangreiche psychophysiologische Versuche entstanden ist.

$$\begin{array}{cccccccc} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{array} \tag{7.3}$$

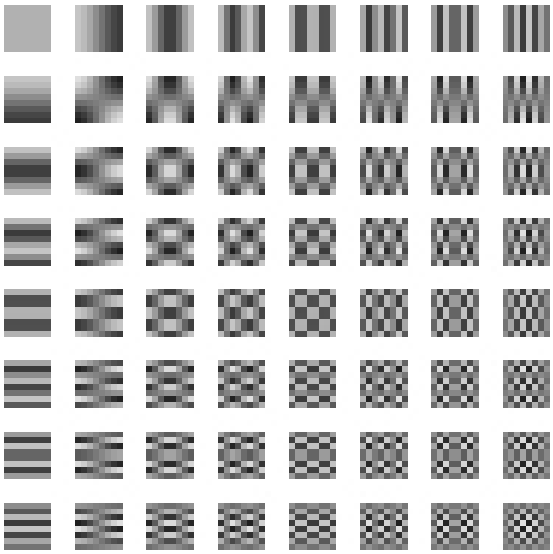


Abbildung 7.2.: Die Basisfunktionen der DCT (Blockgröße 8×8)

7.1. Zickzack-Abtastung

Nachdem die Koeffizienten quantisiert wurden, werden sie in „Zickzack“-Reihenfolge aus dem Block in ein Array übertragen. Dieser Vorgang ist in Abbildung 7.3 dargestellt.

Dadurch wird erreicht, dass in dem Array die Koeffizienten, die am wahrscheinlichsten große Werte haben, am Anfang stehen. Das Array enthält durch die Quantisierung üblicherweise sehr viele Nullen. Diese kann man mit einer Lauflängencodierung zusammenfassen, indem man nicht jede Null einzeln speichert, sondern lediglich, wie viele Nullen nach dieser Stelle folgen. Die „Zickzack“-Abtastung sorgt dabei für eine günstige Anordnung der Nullen im Array.

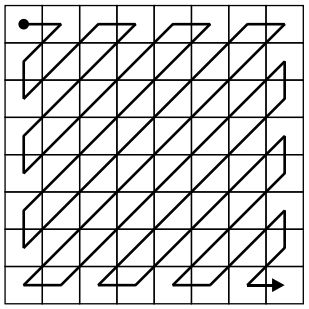


Abbildung 7.3.: „Zickzack“-Abtastung

Abbildung 7.4 zeigt noch einmal zusammenfassend das Blockschaltbild eines Transformationscodierers.

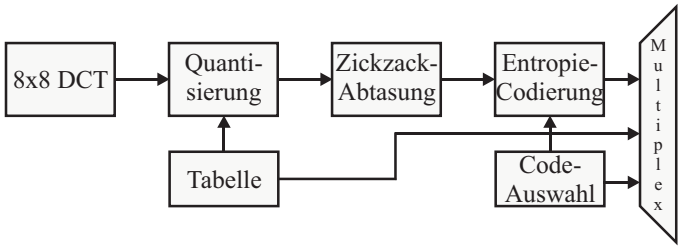


Abbildung 7.4.: Prinzip der Transformationscodierung

8. Hybridcodierung

Bei der *Video*codierung reicht es nicht aus, jedes Bild für sich einer der bisher genannten Arten der Codierung zu unterziehen. Ein Video mit 25 Bildern pro Sekunde und einer Auflösung von $720 \times 576\,\text{Pixel}^2$, wie es in Deutschland für das digitale Fernsehen verwendet wird, hätte unkomprimiert eine Datenrate von ungefähr $125\,\frac{\text{Mbit}}{\text{s}}$. Dabei ist bereits eine 4:2:0-Abtastung berücksichtigt. Für den Videodatenstrom eines Programmes im digitalen Fernsehen steht üblicherweise eine Datenrate von $5\,\frac{\text{Mbit}}{\text{s}}$ zur Verfügung. Das heißt, das Video muss auf etwa 4 % seiner Größe komprimiert werden. Selbst mit einer verlustbehafteten Transformationscodierung ist das nicht möglich.

Es ist also eine intelligente Verknüpfung verschiedener bisher kennengelernter Datenreduktionsmethoden notwendig. Deshalb spricht man auch von *hybrider Codierung*[1].

Abbildung 8.1 (aus [2]) zeigt das Prinzip der hybriden Codierung.

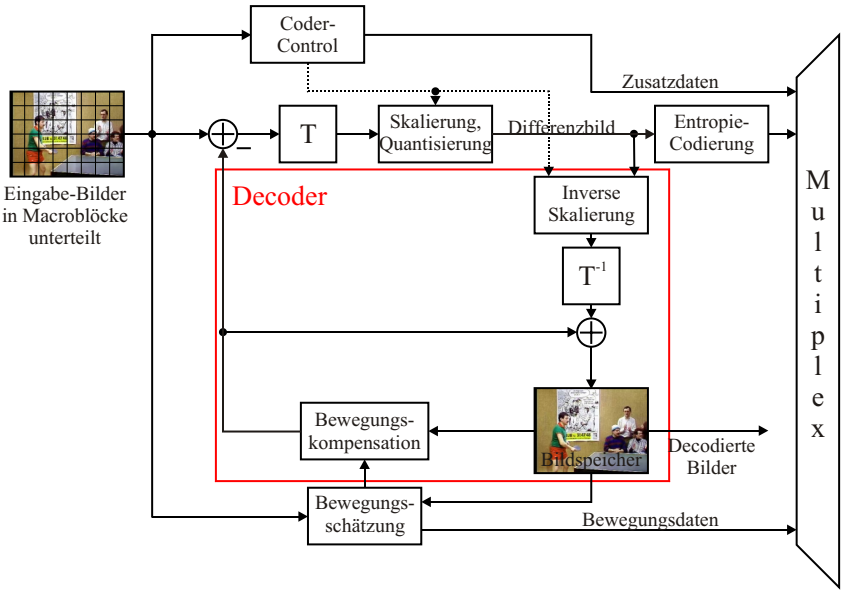


Abbildung 8.1.: Das Prinzip der hybriden Codierung

Für die Prädiktion und die dazugehörige Bewegungskompensation, die aus den Bewegungsvektoren und einem übergebenen Vorbild das aktuelle Bild rekonstruiert, wird das Bild in kleinere, sogenannte *Makroblöcke* zerlegt. Jeder Makroblock erhält einen Bewegungsvektor, der vom aktuellen Block auf einen korrespondierenden Block im Referenzbild zeigt. Der korrespondierende Block muss dabei nicht in das Blockraster passen. Für die Prädiktion werden dann die Pixel des prädizierten Makroblocks auf die Werte des korrespondierenden Blocks gesetzt. Abbildung 8.2 verdeutlicht das Prinzip. Moderne Videocodierverfahren arbeiten mit subpixelgenauen Vektoren. MPEG-2 verwendet zum Beispiel eine Genauigkeit von einem halben Pixel. Die prädizierten Pixel müssen dabei interpoliert werden. Die Größe der Makroblöcke bei MPEG-2 ist 16×16. Der H.264 / AVC-Standard (vergleiche Kapitel 9.2) erlaubt variable Blockgrößen mit einer kleinsten Größe von $4 \times 4\,\text{Pixeln}^2$.

[1] Laut [1] heißt *hybrid* „aus Verschiedenem zusammengesetzt; durch Kreuzung, Mischung entstanden“.

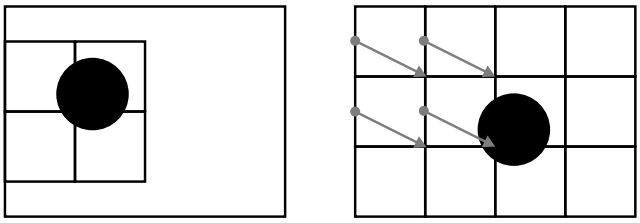


Abbildung 8.2.: Bewegungskompensation; die korrespondierenden Blöcke im linken Bild werden um die grauen Bewegungsvektoren verschoben.

Je kleiner die Makroblöcke gewählt werden, desto besser ist die Prädiktion, aber desto mehr Bewegungsvektoren müssen auch gespeichert werden. H.264 / AVC verwendet deshalb eine ausgefeilte adaptive Codierung, um diese Daten effizient zu speichern. Üblicherweise werden die korrespondierenden Blöcke nach der Methode des kleinsten Fehlerquadrats (MSE[2]) im vorangegangenen Bild gesucht.

Zusätzlich zu dieser einfachen Prädiktion unterstützen moderne Videocodierverfahren die *bidirektionale Prädiktion*. Das bedeutet, dass für die Bewegungskompensation wahlweise Teile eines vorangegangenen Bildes, eines späteren Bildes oder auch eine Linearkombination von beiden verwendet werden können. Das ist sinnvoll, weil in späteren Bildern häufig Inhalte sichtbar sind, die in vorherigen Bildern verdeckt sind, da sich Objekte voreinander bewegen („occlusion-effect"). Aber auch wenn Objekte nicht verdeckt sind, kann oft durch eine bidirektionale Prädiktion ein besseres Ergebnis erzielt werden. Die Reihenfolge, in der die Bilder im Datenstrom erscheinen, muss dafür allerdings umgestellt werden, weil ein reales System immer kausal sein muss. Das heißt, dass Bilder, von denen prädiziert werden soll, im Empfänger schon vorliegen müssen.

Abbildung 8.3 zeigt eine Reihe von Bildern, wie sie üblicherweise in MPEG-2 Videostreams angeordnet sind. Das erste Bild ist immer ein *Intra-Picture* (I-Bild). Ein solches Bild muss vollständig aus sich selbst heraus decodierbar sein. Üblicherweise wird hier eine einfache Transformationscodierung verwendet. Derartige Bilder werden meist in regelmäßigen Abständen in die Sequenz eingefügt, um einen späteren Einstieg in die Sequenz, zum Beispiel bei Fernsehübertragungen, zu ermöglichen. Es folgt ein *P-Picture* („predicted"). Dabei handelt es sich um ein aus dem vorangegangenen I-Bild prädiziertes Bild. Es hat sich eingebürgert, darauf zwei *B-Pictures* („bidirectionally predicted") folgen zu lassen. Das sind Bilder, die aus zwei zuvor decodierten Bildern prädiziert werden, von denen bei MPEG-2 immer eines zeitlich vor und eines hinter dem aktuellen Bild liegt. Die Zahlen in den Bildern, geben die *Presentation-Order* – die Reihenfolge, in der die Bilder abgespielt werden – an. Die Pfeile deuten die Abhängigkeiten an. Der Pfeil von Bild 1 nach Bild 4 bedeutet zum Beispiel, dass Bild 4 aus Bild 1 prädiziert wird. Bild 2 wird teilweise aus Bild 1, teilweise aus Bild 4 prädiziert.

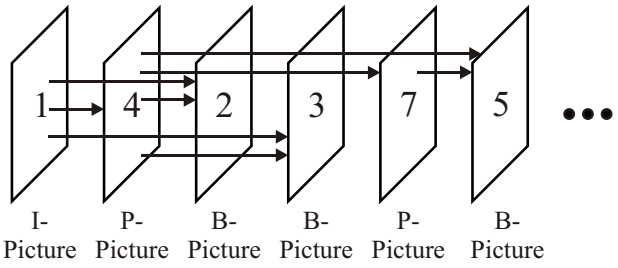


Abbildung 8.3.: Die Reihenfolge der Bilder in einem MPEG-2-Datenstrom

[2] *MSE* steht für *Mean Square Error*

9. Praktische Aspekte der Videocodierung

Mit dem Codierverfahren nach MPEG-2, das zur Zeit das am weitesten verbreitete Verfahren ist, weil es sich als Standard für das digitale Fernsehen durchgesetzt hat, lassen sich Videos in recht guter Qualität mit einer Datenrate von etwa 5 $\frac{\text{Mbit}}{\text{s}}$ codieren. Das führt nicht nur dazu, dass eine Übertragung von digitalen Videos möglich ist, sondern erlaubt sogar, mehrere Programme in einem herkömmlichen Kanal zu übertragen. Damit bietet das digitale Fernsehen gegenüber dem analogen einen klaren Vorteil, da Kanalkapazität ein begrenztes Gut ist.

Inzwischen haben sich die Anwendungsgebiete der Videocodierung aber längst über das klassische Fernsehen hinaus erweitert. Immer kleinere Geräte, wie zum Beispiel Handys, sind im Stande, Videos wiederzugeben oder sogar aufzunehmen, zu versenden und zu empfangen. Videos werden nicht mehr nur über Broadcast-Netze übertragen, sondern auch als Streams im Internet angeboten. Das volle Heimkinoerlebnis soll durch die Einführung von HDTV (High Definition TV), also Fernsehen in höherer Auflösung und besserer Qualität, vermittelt werden. Diese neuen Anwendungen stellen höhere Anforderungen an die Videocodierung als sie MPEG-2 erfüllen kann. Glücklicherweise ist aber auch die Entwicklung der Digitaltechnik vorangeschritten. So kann heutzutage für die Codierung und Decodierung von Videos wesentlich mehr Rechenleistung vorausgesetzt werden, was es ermöglicht, einige der Grundprinzipien der Videocodierung zu verfeinern, wie es in der Vergangenheit schon angedacht, aber mit Rücksicht auf die technische Realisierungsmöglicheiten nicht durchgeführt wurde. Der neueste Standard, H.264 / AVC, stellt im wesentlichen eine derartige Verfeinerung von MPEG-2 dar. Mit H.264 / AVC ist es möglich, die gleiche Qualität bei etwa der halben Datenrate eines MPEG-2-codierten Videos zu erreichen.

Wie man am Beispiel H.264 / AVC erkennen kann, haben die grundsätzlichen Prinzipien der Videocodierung auch über neue Generationen von Codierverfahren hinaus Bestand. Diese Prinzipien sollen in diesem Praktikumsversuch nicht bloß vermittelt, sondern auch praktisch nachvollzogen und auf ihre Effizienz hin untersucht werden. In verschiedenen Aufgaben werden alle Schritte auf dem Weg zu einem codierten Video erarbeitet. Es wird die gesamte Verarbeitungskette von der Vorverarbeitung der Videodaten über die Prädiktion und Transformation bis hin zur Entropiecodierung betrachtet.

9.1. Wo versagt die Hybridcodierung?

In diesem Abschnitt soll kurz darauf eingegangen werden, welche Eigenschaften einer Videosequenz schlecht zu codieren sind und somit zu qualitativ schlechten Ergebnissen bzw. höheren Datenraten führen. Dabei gibt es zwei Komponenten, die zu Problemen führen können: Die Transformationscodierung und die Bewegungskompensation.

Die Transformationscodierung beruht auf der Tatsache, dass die Signalleistung sich auf die niederfrequenten Koeffizienten der DCT oder einer ähnlichen Transformation konzentriert. Dieses trifft im statistischen Mittel zu. Allerdings gibt es auch Bilder bzw. Bildteile, an denen dieses *nicht* zutrifft. Hochfrequente Signalanteile sind zum Beispiel überall dort vorhanden, wo im Bild feine Strukturen oder Kanten auftreten. Auch verrauschte Bilder führen zum Auftreten vieler Koeffizienten (weißes Rauschen ist über alle Koeffizienten gleich verteilt). Dieses führt bei der Codierung dazu, dass an diesen Stellen viele Koeffizienten codiert werden müssen und somit die Datenrate erhöht wird. Im Umkehrschluß gehen bei niedrigen Datenraten – also hoher Quantisierung – Details in diesen Bildteilen verloren. Ein besonders unangenehmer Effekt kann an Kanten

Abbildung 9.1.: Bildausschnitt ohne (links) und mit (rechts) Ringing-Artefakten

auftreten. Da durch die Quantisierung die hohen Frequenzen im Bereich der Kante verloren gehen, entsteht also in diesen Bereichen ein hochfrequentes Fehlersignal, welches sich additiv dem Nutzsignal überlagert. In der Bildcodierung spricht man dort von *Ringing-Artefakten*, da das Fehlersignal aussieht wie Wellen, die sich von der Kante lösen (vergleiche Abbildung 9.1). Im bewegten Fall wird aus dem *Ringing* ein Flimmern, das Ähnlichkeit mit einem Mückenschwarm hat, der um die Kante herum schwirrt. Deswegen spricht man hier vom *Mosquito-Noise*.

Durch die blockweise Verarbeitung der Bilder entstehen zudem zwangsläufig Fehler, die an den Blockkanten besonders stark ausgeprägt sind. Wenn dadurch die Block-Struktur sichtbar wird, spricht man von Block-Artefakten (vergleiche Abbildung 9.2). Diese fallen besonders in Bildbereichen auf, die wenig strukturiert sind. In stark strukturierten Bildbereichen werden diese Artefakte weniger wahrgenommen, weil sie von der Bildstruktur überdeckt werden.

Da *Blockartefakte* als eine der störendsten Fehlerarten ausgemacht wurden, verwenden moderne Codierverfahren, wie zum Beispiel H.264 / AVC, adaptive Filter, um diese zu verringern.

Die Bewegungskompensation trifft ebenfalls Annahmen, die nicht immer eingehalten werden. Durch die blockbasierte Bewegungskompensation werden grundsätzlich zweidimensionale, translatorische Bewegungen beschrieben. Wenn ein Objekt also rotiert oder sich anders als parallel zur Bildebene bewegt, lässt sich seine Bewegung offensichtlich nicht beschreiben. Somit kann die Bewegungskompensation selbst im Idealfall keine perfekte Prädiktion der meisten Bilder liefern. Sind allerdings die Blöcke hinreichend klein, ergibt sich eine gute Approximation. Ebenso treten an Objektkanten Probleme auf, weil hier im Regelfall zwei Bewegungsrichtungen (von zwei Objekten) innerhalb eines Blocks auftreten.

Abbildung 9.2.: Bildausschnitt mit Blockartefakten

Ein weiteres Problem ist, dass mit abgetasteten Bildern gearbeitet wird. Solange die Bilder sauber verarbeitet wurden und dem Shannon'schen Abtasttheorem genügen, ist es trotzdem möglich, jede translatorische Bewegung durch Bewegungsvektoren darzustellen. Dieses ist aber praktisch nie der Fall, da dazu ideale Filter benötigt würden. Ausserdem wird bei der Aufnahme von Bildern und der anschließenden Verarbeitung in dieser Hinsicht gerne auf Kosten der Qualität gespart. Durch die Unterabtastung entstehen, besonders an Kanten, *Alias-Fehler*, was dazu führt, dass diese Bildteile zwangsläufig falsch prädiziert werden. Besonders bei niedrig aufgelösten Sequenzen ergibt sich daraus ein Problem. Beim Herunterrechnen, dem sogenannten *Downsampling*, von Sequenzen

sollte man also darauf achten, dass das Quellbild vor der Unterabtastung passend vorgefiltert wird.

9.2. Aktuelle Verfahren der Videocodierung: H.264 / AVC / MPEG-4 part10

In diesem Abschnitt soll kursorisch der derzeit modernste Standard zur Videocodierung vorgestellt werden, um zu zeigen, wie aktuell die Grundlagen der Videocodierung, die bereits in den 80er-Jahren entdeckt wurden, auch heute noch sind.

Im wesentlichen gibt es zwei Institutionen, die sich mit der Standardisierung von Video-Codierverfahren auseinandersetzen. Auf der einen Seite ist die ISO/IEC JTC1/SC29/WG11 (*International Organization for Standardization / International Electrotechnical Commission, Joint Technical Committee 1, SubCommittee 29, Working Group 11*) zu nennen, die häufiger als *Moving Picture Experts Group* (MPEG) bezeichnet wird. Auf der anderen Seite steht die VCEG (*Video Coding Experts Group*) der ITU-T (der Standardisierungsabteilung der *International Telecommunication Union*). Beide Gruppen haben sich Ende des letzten Jahrtausends zum so genannten *Joint Video Team* (JVT) zusammengeschlossen, um einen Standard zu schaffen, der „doppelt so gut“ wie der MPEG-2-Standard ist, also Videos bei der gleichen Qualität auf die halbe Datenrate komprimiert.

Das Ergebnis der gemeinsamen Arbeit ist der Standard *ISO/IEC 14496-10 „Advanced Video Coding“* (MPEG-4 AVC) oder auch ITU-T Rec. H.264, auch bekannt als H.264 / AVC. Dieser Standard beschreibt ein hybrides Videocodierverfahren. Im Kern arbeitet es so, wie in den vorangegangenen Abschnitten beschrieben. Einige Mechanismen wurden allerdings weiter verfeinert, wodurch tatsächlich eine durchschnittliche Verbesserung der Kompressionsrate auf ungefähr das Doppelte gegenüber MPEG-2 erreicht wird. Die Verbesserung ist möglich, weil heutzutage mehr Rechenleistung für die Codierung und Decodierung von Videos zur Verfügung steht, als es bei der Einführung von MPEG-2 Ende der 80er-Jahre der Fall war.

Während MPEG-2 die Bewegungskompensation mit einer Genauigkeit von einem halben Pixel macht, wobei die Zwischenpixel linear interpoliert werden, verwendet H.264 / AVC eine Genauigkeit von einem viertel Pixel. Eine so exakte Prädiktion macht aber nur dann Sinn, wenn gleichzeitig die Interpolation verbessert wird. Sonst würden Alias-Effekte den Gewinn wieder zunichte machen. Deswegen verwendet H.264 / AVC für die Interpolation ein Filter, das in zwei 6-Tap-Tiefpassfilter separierbar ist, was selbstverständlich wesentlich rechenintensiver ist als die lineare Interpolation (2-Tap-Filter).

Eine weitere Verfeinerung der Bewegungskompensation besteht darin, dass sich jeder 16×16-Makroblock in vier 8×8-Untermakroblöcke zerlegen lässt. Jeder dieser Untermakroblöcke lässt sich erneut in vier 4×4-Blöcke zerlegen. Somit ist eine Bewegungskompensation im kleinsten Fall auf Basis von 4×4-Blöcken möglich, durch die sich Objektkanten und andere nicht prädizierbare Bildbereiche besser approximieren lassen.

Als Referenzbilder stehen bei H.264 / AVC nicht mehr nur ein oder zwei Bilder (für P- oder B-Frames) sondern bis zu 16 Bilder zur Verfügung, von denen aber in praktischen Implementierungen üblicherweise (aus Speichergründen) nur bis zu fünf verwendet werden. Diese Bilder lassen sich fast beliebig verwalten. Das heißt, dass sie nicht unbedingt der Reihe nach verworfen werden müssen. Ein Bild, das sich besonders gut für die Prädiktion der folgenden Bilder eignet, kann über längere Zeit, als sogenanntes *Long-Term-Reference-Picture* im Speicher behalten werden, während seine Nachfolger bereits früher wieder gelöscht werden. Die Prädiktion kann außerdem aus beliebigen Linearkombinationen von vorwärts- und rückwärts-prädizierten Blöcken zusammengesetzt werden.

Auch die Bewegungsvektoren werden nicht einfach gespeichert. Sie werden aus benachbarten Vektoren prädiziert. So ist es bei einigen Blöcken sogar möglich, sie fast ohne jeden Datenaufwand

zu codieren, weil weder ein Differenzbild, noch ein Bewegungsvektor gespeichert werden muss. Lediglich ein Flag muss für jeden Makroblock gespeichert werden, das anzeigt, ob es sich um einen „skipped“ Block handelt (also einen, der keine weiteren Daten benötigt) oder nicht. Die effektive Speicherung der Bewegungsdaten (also der Bewegunsvektoren und Zusatzdaten für die Bewegungskompensation) ist bei H.264 / AVC besonders wichtig, weil mehr Daten gespeichert werden müssen als zum Beispiel bei MPEG-2.

Auch die I-Bilder, also diejenigen Bilder, die komplett aus sich selbst heraus decodiert werden können, werden bei H.264 / AVC prädiziert. Dafür wird eine gerichtete Prädiktion (örtliche Prädiktion) verwendet. Das bedeutet, dass 4×4-Blöcke oder 16×16-Blöcke durch ihre benachbarten Pixel bzw. diejenigen davon, die bereits vorher decodiert wurden, prädiziert werden. Dafür werden jeweils alle Pixel entlang einer Geraden durch dasselbe Randpixel prädiziert.

Für die Transformationscodierung verwendet H.264 / AVC nicht die herkömmliche DCT, sondern eine 4×4-Block-Transformation, die mit reinen Integer-Rechenoperationen auskommt. Im High-Profile, das nachträglich dem Standard hinzugefügt wurde, kann alternativ auch eine 8×8-Block-Integer-Transformation verwendet werden. Diese hat sich besonders bei großen Bildern als effektiver erwiesen.

Um die Bildung der sehr störenden Blockartefakte zu verhindern, verwendet H.264 / AVC ein Deblocking-Filter. Dabei handelt es sich um ein adaptives Filter, das an den Rändern der Transformationsblöcke nach Kanten sucht, die für Block-Artefakte typisch sind, und diese gegebenenfalls weichzeichnet. Dieses Filter ist ein *In-Loop-Filter*, was bedeutet, dass das Bild gefiltert wird, bevor es für die Prädiktion weiterer Bilder verwendet wird. Im Gegensatz dazu verwenden viele Codechersteller[1] Nachverarbeitungsfilter, die zwar weniger effektiv sind, aber dadurch, dass sie nicht in die Decoder-Schleife eingreifen, beliebig angepasst werden können, so dass der Video-Datenstrom standardkonform bleibt.

Für die Entropiecodierung verwenden praktisch alle älteren Standards eine VLC-Codierung. Dabei werden üblicherweise mehrere feste Codetabellen zur Verfügung gestellt, von denen dann für jeden Fall die beste ausgewählt werden kann. H.264 / AVC bietet zwei Möglichkeiten für die Entropiecodierung. Beide arbeiten adaptiv. Das bedeutet, dass ein Codewort nicht nur vom zu codierenden Symbol, sondern auch von vorher codierten beziehungsweise decodierten Symbolen abhängt. Die eine Variante verwendet dafür größere Tabellen mit VLCs. Diese Variante nennt sich *Context based Adaptive Variable Length Coding* (CAVLC). Die zweite Variante, *Context based Adaptive Binary Arithmetic Coding* (CABAC) verwendet eine arithmetische Codierung mit einem ausgeklügelten und sehr rechenintensivem adaptiven Wahrscheinlichkeitsmodell.

9.3. „Profiles“ und „Levels“

Weil es nicht für jede Anwendung nötig ist alles zu unterstützen, bieten die wichtigsten Codierstandards die Möglichkeit, standardkonforme Decoder zu bauen, dabei aber nur eine Untermenge des Standards zu unterstützen. Derartige Untermengen werden durch die so genannten *Profiles* und *Levels* angegeben.

Profiles definieren Untermengen der **algorithmischen Möglichkeiten** eines Standards, während *Levels* der **Einschränkung bestimmter Parameter** dienen. So gibt es bei *MPEG-2* zum Beispiel ein *Profile*, das eine 4:2:2-Abtastung der Chrominanz-Komponenten erlaubt, während das so genannte *Main-Profile*, das von DVB für die Übertragung vorgeschrieben wird, eine 4:2:0-Abtastung verwendet. Auch die Verwendung der bidirektionalen Prädiktion kann durch die Wahl eines entsprechenden Profiles ausgeschlossen werden. Die *Levels* schränken bei MPEG-2 die Auflösungen und Datenraten ein. Das Main-Level erlaubt beispielsweise Auflösungen bis 720×576 Pixeln bei Datenraten von bis zu $15\,\frac{\text{Mbit}}{\text{s}}$, das *High-Level* erlaubt Auflösungen bis zu 1920×1152 Pixel[2] bei Datenraten bis $80\,\frac{\text{Mbit}}{\text{s}}$. Abbildung 9.3 zeigt die *Profiles* und *Levels* des

[1] *Codec* ist ein Kunstwort aus den beiden Worten (En-)*Co*der und *Dec*oder.

MPEG-2-Standards, die tatsächlich angewendet werden. Wenn man eine bestimmte Kombination aus *Profile* und *Level* meint, schreibt man üblicherweise so etwas wie *MP@ML*, und meint damit *Main-Profile at Main-Level*.

Level \ Profile	Simple	Main	4:2:2 Profile	SNR	Spatial	High
High		4:2:0 1920x1152 80 Mbit/s I, P, B	4:2:2 1920x1152 300 Mbit/s I, P, B			4:2:0 1920x1152 100 Mbit/s I, P, B
High-1440		4:2:0 1440x1152 60 Mbit/s I, P, B			4:2:0 1440x1152 60 Mbit/s I, P, B	4:2:0 1440x1152 80 Mbit/s I, P, B
Main	4:2:0 720x576 15 Mbit/s I,P	4:2:0 720x576 15 Mbit/s I, P, B	4:2:2 720x576 50 Mbit/s I, P, B	4:2:0 720x576 15 Mbit/s I, P, B		4:2:0 720x576 20 Mbit/s I, P, B
Low		4:2:0 352x288 4 Mbit/s I, P		4:2:0 352x288 4 Mbit/s I, P, B		

Abbildung 9.3.: Profiles und Levels bei MPEG-2

Die Level-Struktur bei *H.264/AVC* (siehe auch Kapitel 9.2) ist ein wenig komplizierter als die bei MPEG-2. Sie dient aber denselben Zielen und ist in Abbildung 9.4 illustriert.

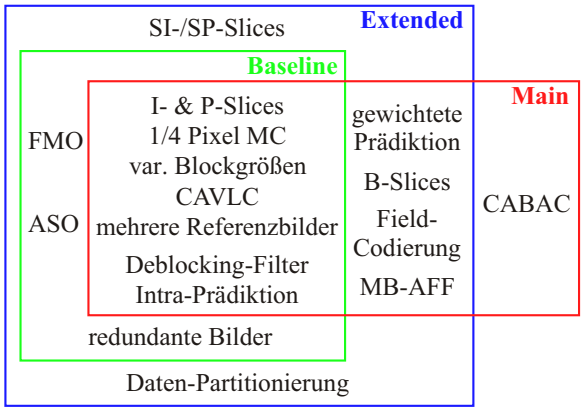


Abbildung 9.4.: Profiles bei H.264/AVC

10. Qualitätsvergleich von Videos

In den vergangenen Abschnitten war häufiger die Rede davon, dass ein Codec beispielsweise *doppelt so effektiv* sei wie ein anderer. Um solche Aussagen treffen zu können, müssen die betrachteten Codes zunächst anhand eines allgemein anerkannten Maßes verglichen werden. Die vorherrschende Meinung ist, dass als bestes Maß dafür die Meinung des Betrachters in Frage kommt. Diese zu ermitteln und zu einem brauchbaren Meßwert zusammenzufassen ist allerdings weniger leicht, als es den Anschein haben mag.

Zunächst einmal wird nicht jeder Betrachter die Qualität eines Bildes oder einer Bildsequenz gleich bewerten. Aus diesem Grund muss eine repräsentative Gruppe von Betrachtern befragt werden. Die Meinungen all dieser Betrachter werden dann im so genannten *Mean Opinion Score* (MOS) gemittelt und bei seriösen Tests mit einem Konfidenzintervall versehen.

Weiterhin spielen Faktoren wie die Qualität und Art des Displays, die Umgebungsbeleuchtung, der Betrachtungsabstand und die Größe des Displays eine Rolle bei der Bewertung der Qualität. Um trotz dieser Einflüsse Tests zu ermöglichen, deren Ergebnisse reproduzierbar und allgemein anerkannt sind, werden in der ITU-R Rec. BT.500 Testverfahren beschrieben, die für praktisch alle derartigen Tests angewendet werden. Dabei werden sämtliche Parameter festgelegt, die die Bewertung der Qualität beeinflussen können.

Solche so genannten *subjektiven Tests* sind mit einem hohen zeitlichen - und damit auch finanziellen - Aufwand verbunden. Aus diesem Grund wurden in der Vergangenheit verschiedene Versuche unternommen, den MOS durch objektive Verfahren, also Werte, die ohne die Befragung von Testpersonen aus dem Bildmaterial berechnet werden können, zu ersetzen. Ein sehr einfaches Beispiel für solch ein Verfahren ist der *PSNR* (Peak Signal to Noise Ratio). Dieser Wert berechnet sich zu

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{(\text{max. Signalleistung})^2}{\mathrm{MSE}} \right), \tag{10.1}$$

wobei MSE den mittleren quadratischen Fehler bezeichnet. Man kann ihn interpretieren als die maximal mögliche Signalleistung, bezogen auf die Leistung des Fehlers.

Dieser Wert berücksichtigt zwar nicht die komplizierten Zusammenhänge des menschlichen Sehsinnes, ist aber für den Vergleich verschiedener Codierverfahren und ähnliche Anwendungen allgemein anerkannt. Solange die verglichenen Bilder gleichartige Fehler beinhalten, also lediglich die Stärke eines auftretenden Fehlers verglichen werden soll, ist dieses sicherlich auch zulässig. Allerdings ist beim Umgang mit einem solchen Qualitätsmaß grundsätzlich Vorsicht geboten. Ein leichtfertiges Gleichsetzen mit *der Videoqualität* ist streng genommen nicht zulässig.

11. Ausblick

Am Schluß des theoretischen Teils soll mit einem Ausblick geschlossen werden, wohin die Reise in der Videocodierung gehen könnte:

Dabei zeichnen sich einige Ziele in nicht allzuferner Zukunft ab, deren Erreichen und vor allem deren Durchsetzung im Massenmarkt zwar sehr ungewiss sind, jedoch derzeit von der Industrie und der Filmbranche fokusiert werden.

Zum einen wird angestrebt, das herkömmliche Kino durch ein digitales Pendant zu ersetzen, was sicherlich nicht zuletzt wegen des wirtschaftlichen Einsparpotentials für die Filmindustrie interessant ist. Zudem lässt sich die Distribution des Filmmaterials in digitaler Form weitaus schneller bewerkstelligen als analoge Kopien der Filmrollen zu erstellen und weltweit an die Kinos zu schicken. [7]

Zu guter Letzt jedoch ist nach [6] auch denkbar, dass die derzeitige Auflösung, die im analogen Kino durch die Qualität des Filmmaterials für einen Breitbandfilm auf ca. 20.000.000 $\frac{\text{Bildpunkte}}{\text{Einzelbild}}$ beschränkt ist, durch Digitaltechnik in weitaus höhere Auflösungsregionen getrieben wird. So sind für Standbilder im medizinischen Bereich schon 1995 100 $\frac{\text{Megapixel}}{\text{Einzelbild}}$ möglich gewesen. Die dabei anfallende Datenrate wäre jedoch (unkomprimiert) heute noch nicht zu bewältigen, würde sie doch einige 100 $\frac{\text{Gigabit}}{\text{s}}$ betragen.

Die zweite Zukunftsvision besteht nach [6] in der Erschließung der dritten Dimension. Schon seit Jahrzehnten sind Methoden bekannt, mit denen entweder durch Rot-Grün-Brillen oder aber Polarisationsbrillen beiden Augen leicht unterschiedliche Bilder präsentiert werden. Das Gehirn ermittelt dann aus den zwei Wahrnehmungen der Augen ein dreidimensionales Bild, ähnlich des realen Stereosehens, was bekanntlich ebenfalls durch leicht versetzte Bildsignale der beiden Augen funktioniert.

Der Nachteil dieser Verfahren ist jedoch zum einen, dass durch die ungewohnte und nicht reale Tiefenwahrnehmung häufig Kopfschmerzen als Folgeerscheinung auftreten, zum anderen natürlich auch, dass Verfahren, bei denen zusätzliche Hilfsmittel wie speziell ausgestattete Brillen nicht für den täglichen Fernsehgenuß geeignet sind.

In jüngster Zeit gibt es verschiedene neue Ansätze wie den von Philips 3D Solutions seit vier Jahren vorangetriebenen Ansatz der autostereoskopischen Displays, bei dem keine Brille mehr notwendig ist, da die unterschiedlichen Bilder durch eine sogenannte Lentikularlinsenmatrix zu den Augen gelenkt werden. Der Effekt ist ähnlich wie bei den bekannten Wackelbildern.

Noch 2008 hat Philips erste Geräte mit hoher Auflösung ($4096 \times 2160\,\text{Bildpunkte}^2$) vorgestellt, im April 2009 jedoch überraschenderweise die Auflösung der 3 D-Sparte aus wirtschaftlichen Gründen angekündigt. Zwar glaube der Mutterkonzern nach wie vor an eine bevorstehende Wende hin zum dreidimensionalen Fernsehen, diese werde allerdings noch eine nicht näher genannte Zeit auf sich warten lassen.

Insofern bleibt auch die genaue Zukunft der Videocodierung ungewiss. Eines allerdings steht heute schon fest: Gleich, welche Technik sich in Zukunft etablieren wird, ohne Videocodierung ist eine Gesellschaft mit immer weiter voranschreitenden digitalisierten Medien nicht mehr denkbar! [3]

Teil II.

Hausaufgaben

12. Erste Hausaufgabe: Huffman-Codierung

a) **Berechnen Sie die Auftrittswahrscheinlichkeiten aller Symbole der Zeichenkette: »In Ulm um Ulm und um Ulm herum.«** Der Rechenweg soll nachvollziehbar aufgezeichnet werden, damit bei etwaigen Problemen Hilfestellungen durch die Übungsleiterin bzw. den Übungsleiter gegeben werden können. Sie können sämtliche Aufgaben direkt auf dem Aufgabenblatt in dem jeweils unter den Fragen vorgesehenen Platz lösen. Verwenden Sie bei Bedarf ein gesondertes Beiblatt!

> **Achtung:** Leerzeichen und Satzzeichen (Punkte) sind ebenfalls Symbole. Betrachten Sie zudem große und kleine Buchstaben als extra Symbole!

b) **Berechnen Sie die Entropie der gesamten Zeichenkette.**

c) **Erstellen Sie einen optimalen VLC für diese Zeichenkette nach dem Prinzip der**

Huffman-Codierung, und zeichnen Sie den Codebaum!

13. Zweite Hausaufgabe: örtliche Prädiktion

Sie sehen in Abbildung 13.1 ein Bild, welches mit einer Quantisisierung von 2 Bit graustufencodiert ist.

3	3	2	2	2	2	2	2
3	3	2	0	2	2	2	2
2	2	0	0	0	2	2	2
1	1	1	0	1	1	1	1
1	1	0	1	0	3	3	1
1	1	1	1	1	3	3	1

Abbildung 13.1.: Hausaufgabe zur örtlichen Prädiktion: Differenzcodierung

a) **Berechnen Sie die Entropie dieses Bildes!** Der Rechenweg muss nachvollziehbar zu Papier gebracht sein, damit die Betreuerin bzw. der Betreuer bei einem falschen Endergebnis Hilfestellungen zur Berechnung der Entropie geben kann.

b) **Codieren Sie das Bild als Differenzbild mithilfe der Methode der örtlichen Prädiktion!** Nehmen Sie Nullen (**nicht** Einsen!) links außerhalb des Bildes an um die Differenz zu den ersten Randpixeln codieren zu können. Codieren Sie bitte zeilenweise und prädizieren lediglich aus dem direkten linken Nachbarn. Falls Sie weitere Hilfestellungen benötigen, schlagen Sie bitte im Kapitel 6.1 nach. Sie können Ihr Ergebnis in die abgedruckte Leertabelle 13.1 eintragen.

c) **Wie groß ist der Wertebereich des Bildes *vor* der Codierung?**

Tabelle 13.1.: Leertabelle für Hausaufgabe zur örtlichen Prädiktion

d) **Wie groß ist der Wertebereich des Differenzbildes (*nach* der Codierung)? Was fällt Ihnen auf? (Ist der neue Wertebereich größer oder kleiner geworden und warum?)**

e) **Mit welcher Technik lässt sich trotz des veränderten Wertebereiches eine Datenrateneinsparung erreichen?** (Gemeint ist ein grundlegendes Verfahren, welches im Zusammenhang mit örtlicher Prädiktion in der Praxis eingesetzt wird und nicht weiterführende Methoden wie etwa Transformationscodierung.)

Teil III.

Aufgaben & Protokolle

14. Aufgabenstellungen für die Versuche

Im praktischen Teil des Versuches soll Schritt für Schritt ein hybrides Codierungssystem aufgebaut werden. Hierbei können die im Theorieteil erworbenen Kenntnisse angewandt und vertieft werden. Natürlich kann in dem begrenzten Zeitrahmen nur ein relativ simples System aufgebaut werden. Allerdings sind die dabei gemachten Beobachtungen auch für reale System gültig und sehr hilfreich um Inhalte von vorangegangen oder vertiefenden Vorlesungen wie *Bildkommunikation II* besser zu verstehen.

Zur einfacheren Lesbarkeit werden für die Versuchsbeschreibungen folgende Konventionen eingeführt:

- *Programm-, Datei- sowie Eigennamen von Programmbestandteilen sind in Schrägschrift,*
- `Befehle, die eingegeben oder Buttons, die angeklickt werden müssen, in` *`Typewriter`*`,`
- Tasten, die gedrückt werden müssen in Kapitälchen,
- **sowie Pfadangaben fettgedruckt**

gesetzt.

Zu jedem Versuch ist ein Protokoll anzufertigen. Diese Aufzeichnungen verbleiben zum späteren Nachschlagen in den Unterlagen, d. h., sie müssen nicht abgegeben werden. Während der einzelnen Versuchsabschnitte sollten dennoch Notizen gemacht werden, damit in den Besprechungen mit der Betreuerin oder dem Betreuer **während** und **nach** den Versuchen eine qualifizierte Analyse genommen werden kann.

Natürlich steht es frei, eigene Protokolle zu erstellen, jedoch wird empfohlen, die Vordrucke zu verwenden, die im Abschnitt 15 angehängt sind. Bei den Versuchen wird an geeigneter Stelle auf entsprechend vorbereitete Tabellen verwiesen.

14.1. Die Simulationsumgebung

Die im Folgenden beschriebenen Versuche basieren auf der Simulationsumgebung *Simulink*, die als Bestandteil der Matlab-Suite von The MathWorks vertrieben wird. In diesem Versuch wird die Version *R*2014*a* eingesetzt.

14.2. Versuchsvorbereitung

Zu Beginn des Versuches muss das Programm Matlab auf dem Testrechner gestartet werden. Dieses geschieht entweder über das entsprechende Symbol im Startmenü oder auf dem Desktop oder direkt über `START` → `Ausführen...` und Eingabe von `matlab` gefolgt von einem Druck auf die Eingabe-Taste.

Als *Current Folder* im Dateifenster auf der linken Seite muss der Ordner **C:\Praktikum\simulink** eingestellt werden. Sollte das Dateifenster nach Start der Matlabprorammoberfläche gar nicht erscheinen, kann es über den Befehl `filebrowser` im *Command Window* eingeblendet werden.

Stellen Sie als nächstes sicher, dass unter `ENVIRONMENT` → `Set paths...` das Verzeichnis **C:\Praktikum\simulink\videocodierung_library** eingestellt ist. Sollte dieses nicht der Fall sein, ergänzen Sie es mittels `Add folder...`. Die Sicherheitsabfrage beim Schließen des Dialogs können Sie getrost verneinen, da Sie mangels Berechtigung ohnehin die Pfadangaben nicht im Programme-Ordner speichern dürfen.

Wichtig: Die Pfadangaben und das aktive Verzeichnis müssen **auf jeden Fall** stimmen, da ansonsten die Simulinkmodelle zu den Versuchen nicht lauffähig sind!

Als nächstes wird im *Command Window* am Eingabeprompt `simulink` eingegeben. Es erscheint die *Simulink Library* mit verschiedenen in sogenannte *Bibliotheken* geclusterten Ausklappmenüs auf der linken Seite. Die Bibliotheken sind thematisch und nach *Toolboxen* sortiert.

Die Toolbox, mit der Sie sich in diesem Versuch beschäftigen werden, heißt *Videocodierung* und ist alphabetisch einsortiert, weshalb sie relativ weit unten in der *Simulink Library* zu finden ist.[1]

14.3. Betrachtung des Testmaterials

Zeitvorschlag: 5 min

Bevor Sie mit den eigentlichen Aufgaben beginnen, sollen Sie in diesem vorbereitenden Abschnitt lediglich das Testmaterial in Augenschein nehmen. Das Videomaterial liegt als unkomprimierte Rohdaten im Format 352×288 (CIF) vor. Die Bildwiederholfrequenz beträgt $25\ \frac{\text{Vollbilder}}{\text{s}}$.

Öffnen Sie die Datei *quellmaterial_anschauen.mdl* aus dem Ordner **simulink** und klicken Sie doppelt auf den Block *Videodatei anschauen*. Sie sehen die Parametereinstellungen im unteren Eingabefeld des Blockes. Einzelne Übergabewerte sind durch Kommata »,« getrennt. Die Bedeutung der Einstellungen sind wie die aller anderen Blöcke ebenfalls im Anhang A aufgelistet, sollen hier exemplarisch allerdings noch einmal erläutert werden:

Der Reihe nach haben die Parameter folgende Bedeutung: YUV-Videodatei in Einzelhochkomma »'«, Videobreite in Pixeln, Videohöhe in Pixeln, Nummer des ersten abzuspielenden Frames, Nummer des letzten abzuspielenden Frames, Framedatenrate in $\frac{\text{Frames}}{\text{s}}$, Anzahl der Abspielwiederholungen, Nummer des Videofensters, vertikale Position des Videofensters[2], horizontale Position des Videofensters[2].

Es sollten Parameter wie folgt oder ähnlich eingestellt sein: `'../videos/susie.yuv', 352, 288, 1,089, 25, 2, 3, 150, 150`. Der Videoabschnitt, der Ihnen bei dieser exemplarischen Einstellung angezeigt wird, beginnt beim ersten Bild und endet beim 89ten. Das Video hat eine Gesamtlänge von 749 Bildern. Aufgrund der internen Videobehandlung von MATLAB würde allerdings die Verarbeitung des gesamten Videos abhängig von der Ausstattung des Testrechners eine lange bis sehr lange Zeit in Anspruch nehmen. Beschränken Sie sich also im eigenen Interesse auf das Betrachten eines Teilbereiches bzw. schauen Sie sich das Gesamtmaterial in Abschnitten von ca. $100-200$ Bildern an. Ihr Betreuer wird Ihnen Hinweise zu sinnvollen Einstellungen geben!

Starten Sie nun die Simulation, indem Sie den `Play-button` in der Menüleiste drücken und betrachten Sie das Testvideo, mit dem Sie den restlichen Versuch über arbeiten werden. Bedenken Sie, dass auch kurze Videostücke erst von MATLAB verarbeitet werden müssen und gedulden Sie sich dementsprechend bis zum Start des Videos einige Zeit!

[1] **Hinweis:** Die *Videocodierung*-Toolbox ist eine eigens für diesen Versuch entwickelte und zusammengestellte Toolbox. Sie werden diese also nicht auf anderen MATLAB-Installationen vorfinden.

[2] **Hinweis:** In MATLAB beginnt die Zählung der Bildschirmkoordinaten am unteren linken Bildschirmrand mit $(0, 0)$. Die erste Koordinate kennzeichnet die horizontale Richtung, die zweite die vertikale.

Tipp: Eine Simulation kann durch Drücken der Tastenkombination STRG + T gestartet werden und mit dem gleichen Befehl auch abgebrochen werden. Dazu muss das Modellfenster zur Zeit des Abbruchkommandos angewählt sein.

Achten Sie beim Begutachten der Szene besonders auf feine, veränderliche Bildstrukturen (beispielsweise die Haare) und auf homogene Flächen (wie etwa den Hintergrund). Schauen Sie die Testsequenz gegenebenfalls mehrfach an. Sie können anhand des sechsten Parameters die Anzahl der automatischen Wiederholungen der Szene einstellen.

14.4. Erste Aufgabe: Örtliche Prädiktion und Entropie

Zeitvorschlag: 40 min

Der erste Versuch beschäftigt sich mit dem Thema der örtlichen Prädiktion. Laden Sie zu Beginn die Datei *oertliche_praediktion_student.mdl* aus dem Ordner **simulink**. Sie finden in der vorbereiteten Datei bereits die Blöcke zum Einlesen der Einzelbilder aus dem Videostream sowie die Clock, die die aktuelle Bildnummer anzeigt. In der Mitte des Arbeitsblattes ist bereits ein Messkanalmodell eingefügt. Es handelt sich um den großen Block, welcher sich etwa in der Mitte des *Simulink*-Modells befindet. Da der Messkanal für alle Versuche der gleiche sein wird, hat er bereits alle Ein- und Ausgänge, auch wenn an dieser Stelle noch nicht alle benötigt werden.

Tipp: Markierte Blöcke können mit STRG + I um 180° gedreht werden, eine Rotation um 90° im Uhrzeigersinn ist mit der Tastenkombination STRG + R möglich.

Bevor Sie mit dem Aufbau des Prädiktors beginnen, überprüfen Sie zunächst die Einstellungen unter `Simulation` → `Configuration Parameters`. Sie können diesen Dialog ebenfalls durch Drücken der Tastenkombination STRG + E erreichen. Die Einstellungen sollten wie in Abbildung 14.1 abgebildet eingestellt sein. Sollten die Auswahlboxen aus irgendwelchen Gründen andere Werte enthalten, korrigieren Sie diese entsprechend.

Die Werte bei *Start Time* und *Stop Time* kennzeichnen die Bildnummern, die beim Starten einer Simulation aus der Datei, welche im Block *Lese Bild aus Datei* eingestellt ist, verarbeitet werden. Sie dürfen diese gerne beliebig verstellen um sich verschiedene Teile des Videos anzuschauen. Bedenken Sie bei der Wahl eines Intervalls, dass in der Simulation Einzelbilder angezeigt werden und zwischen jeweils zwei Bildern eine gewisse Verarbeitungszeit vergeht! Das Beispielvideo *'susie.yuv'* hat 749 Bilder. Sollten Sie größere Werte für die Bildnummer einstellen, werden Sie eine Fehlermeldung erhalten, dass die spezifizierte Stelle in der Datei nicht gelesen werden kann.

Tipp: Um zwei Blöcke zu verbinden, kann entweder mit der linken Maustaste der Ausgang des einen Blocks angeklickt werden, die Maustaste gedrückt gehalten und der Eingang des Folgeblockes angewählt werden oder aber der Startblock durch einmaliges kurzes Anklicken aktiviert werden und bei gedrückt gehaltener STRG-Taste mit der linken Maustaste auf den Zielblock geklickt werden. Letzteres verbindet jedoch *stets* den ersten freien Ausgang des Startblocks mit dem ersten freien Eingang des Zielblocks und ist daher nicht immer von Nutzen.

Damit Sie die vom Messkanalmodell ermittelten und am entsprechenden Ausgang zur Verfügung gestellten Entropiewerte notieren können, benötigen Sie den *Entropie*-Block[3] aus der *Videocodierungsbibliothek*, der bereits an das Messkanalmodell angeschlossen sein sollte. Die

[3] Es handelt sich um ein gewöhnliches *Display*-Element, welches speziell zur Entropieanzeige in die *Videocodierungsbibliothek* aufgenommen worden ist.

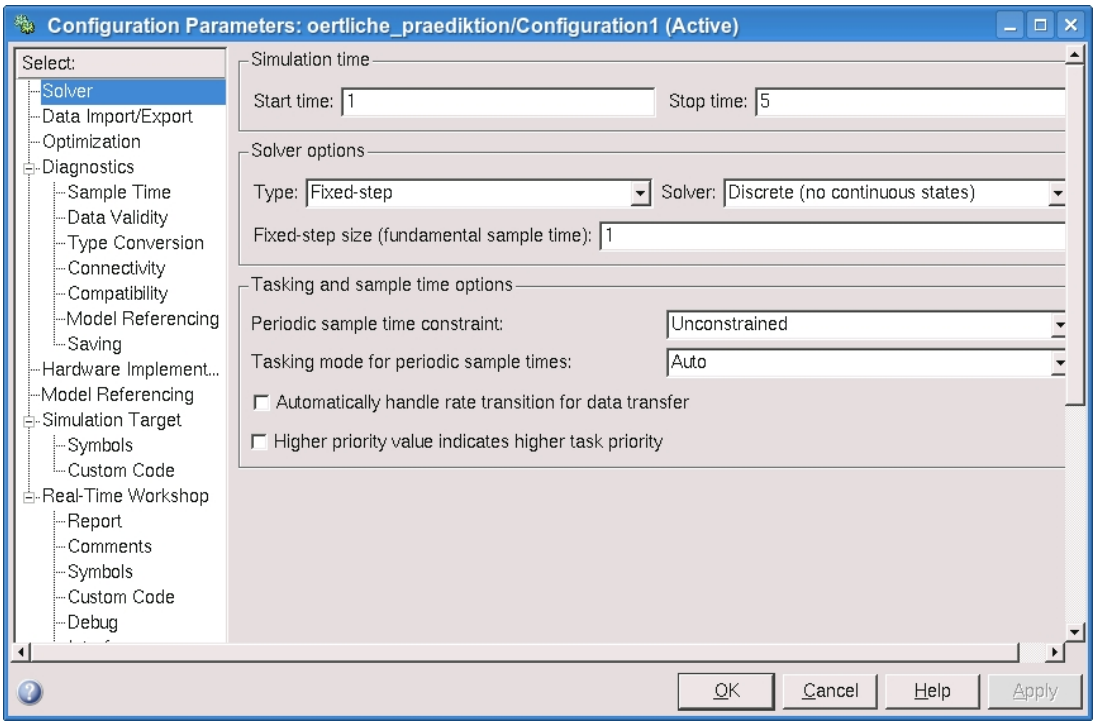


Abbildung 14.1.: *Configuration Parameters* zum Versuch örtliche Prädiktion

ebenfalls bereits vorhandenen Anzeigeblöcke *zeige Bild in Fenster <x>* und *Histogramm in neuem Fenster anzeigen* haben zwar keinen funktionalen Einfluss auf den Simulationsablauf, sind jedoch für die Anschauung und die Beantwortung einiger Fragen zweckdienlich.

> **Tipp:** Um eine Abzweigung von einer bereits existierenden Leitung zu erstellen, muss diese zunächst durch einfaches Anklicken markiert werden. Anschließend muss die STRG-Taste gedrückt gehalten werden und währenddessen mit der linken Maustaste von einem Knotenpunkt der Verbindungslinie aus eine weitere Linie gezogen werden.

Überprüfen Sie ob die Blöcke korrekt angeordnet und verbunden sind. Anschließend müssen Sie die benötigten Parameter einstellen. Doppelklicken Sie dazu grundsätzlich auf einen Block und geben im Textfeld *Parameters* die entsprechenden Werte ein. Die Anzahl der Parameter, ihre Bedeutung sowie empfohlene Werte entnehmen Sie dem Kapitel A. An dieser Stelle genügt es, alle eingestellten Parameter zu überprüfen. Die Einstellungen sollten bereits korrekt sein. Korrigieren Sie fehlerhafte Einstellungen entsprechend.

Nachdem Sie mit dem Aufbau des Modells fertig sind, versuchen Sie wieder mit dem `Play-button` bzw. STRG + T die Simulation zu starten.

> **Tipp:** In der Regel werden Ihnen die Blöcke bei formal nicht korrekten Parametern eine Fehlermeldung liefern und Hinweise zum korrekten Eingabeformat liefern. Lesen Sie die auf den ersten Blick kryptisch anmutenden Fehlermeldungen aufmerksam durch, Sie werden Hinweise zur Fehlerbehebung finden!

Ihre Betreuerin oder Ihr Betreuer wird Ihnen helfen, falls Sie Probleme mit dem Versuchsaufbau oder Starten der Simulation haben sollten.

Fragen zur Simulation

Nachdem Sie die Simulation zum Laufen gebracht haben, beantworten Sie untenstehende Fragen und vermerken Ihre Antworten in einem Protokoll. Sie können dazu den Vordruck im Kapitel 15.1 verwenden.

Besprechen Sie anschließend Ihre Ergebnisse mit Ihrer Betreuerin bzw. Ihrem Betreuer.

1.) **Ihr Quellmaterial weist eine 8 bit Quantisierung auf. Welche Entropie erwarten Sie in diesem Videomaterial *ohne* jegliche Prädiktion?** (Die Angabe einer Größenordnung und des Maximalwertes genügen!)

2.) **Überprüfen Sie Ihre Vermutung, indem Sie das Bildmaterial direkt von der Quelle in den Eingang *Differenzbild* des Messkanalmodells (oberster Eingang des großen Blockes *Messkanal*) einkoppeln[4]und am Ausgang mit dem *Display*-Element *Entropie* ebenjene ablesen. Ermitteln Sie anhand der ersten fünf Bilder den arithmetischen Mittelwert und notieren diesen!**

> **Tipp:** Der *Messkanal* gibt zusätzlich die Entropie im *Command Window* von MATLAB aus. Somit können Sie sich das notieren ersparen. Durch eingabe des Befehls `mean([wert1,wert2,...])` in der Kommandozeile können Sie sich den Mittelwert ihrer Werte bequem von MATLAB berechnen lassen.

Aktivieren Sie nun wieder den Prädiktor, indem Sie die entsprechenden Verbindungen wiederherstellen!

> **Tipp:** Stellen Sie durch Doppelklick auf den *oertlichen Prädiktor*-Block sicher, dass Sie die korrekte Prädiktionsmatrix eingestllt haben. Die Prädiktionsmatrix ist der zweite Parameter des Blockes. Er sollte auf `[0 0.5 0; 0.5 0 0]` eingestellt sein. Ansonsten korrigieren Sie die Einstellung entsprechend!

3.) **Was wird in den sich öffnenden Fenstern dargestellt? Was sagt das Histogramm am Anfang des Systems (direkt nach dem Auslesen aus der Datei) aus? Was wird überhaupt dargestellt und wie beurteilen Sie die Anzeige verglichen mit dem Histogramm direkt hinter dem Messkanal? Was wird in letzterem Histogramm dargestellt?** Zur Identifikation der Fenster vergeben Sie in den Einstellungen der Anzeigeblöcke eindeutige Nummern. (Es genügen qualitative Beschreibungen und Analysen!)

4.) **Was passiert, wenn Sie im senderseitigen Prädiktor eine andere Prädiktionsmatrix vorgeben als empfängerseitig?** Probieren Sie beispielsweise die Werte `[0 0.2 0; 0.8 0 0]` für den senderseitigen Prädiktor und `[0 0.5 0; 0.5 0 0]` auf der Empfangsseite aus, damit der Effekt überdeutlich hervortritt!

5.) **Wie groß wäre die Datenrate in einem System mit idealer Entropiecodierung für folgende Fälle: Sie möchten**

- **8 bit quantisiertes Quellmaterial *ohne* Prädiktionscodierung übertragen.**
- **8 bit quantisiertes Quellmaterial *mit* Prädiktionscodierung übertragen.**

Ermitteln Sie die benötigten Datenraten anhand voriger Ergebnisse aus diesem Versuch! Achten Sie auf die Angabe einer Einheit, da es ansonsten leicht zu Missverständnissen kommen kann! Die Berechnung mit dem Durchschnitt Ihrer Werte genügt, Sie müssen keine bildgenaue Berechnung durchführen. (Hinweis: Das Quellmaterial hat eine Auflösung von 352 Pixeln in horizontaler und 288 Pixeln in vertikaler Richtung, das Material hat eine Bildwiederholfrequenz von 25 $\frac{\text{Bildern}}{\text{s}}$.)

[4] Die Beschriftung ignorieren Sie an dieser Stelle erst einmal! Der Einfachheit der Versuchsdurchführung halber wurde auf die Erstellung unterschiedlicher Messkanalmodelle verzichtet.

14.5. Zweite Aufgabe: Transformationscodierung

Zeitvorschlag: 30 min

Analog zu dem bisherigen örtlichen Prädiktionssystem ist in der Datei *'transformationscodierung_student'* im Ordner **simulink** eine senderseitige Transformationscodierung und empfängerseitige -decodierung realisiert.

Es wurde im theoretischen Teil bereits erklärt, dass eine Transformationscodierung normalerweise blockweise arbeitet. Deshalb werden im Simulinkmodell blockorientierte Verarbeitungsalgorithmen eingesetzt.

Sie sollen in dieser Aufgabe die Funktionsweise einer solchen Verarbeitung und den Einfluss verschiedener Quantisierungsmatrizen erleben.

Fragen zur Transformationscodierung

1.) Sie können in den Simulationsblöcken *Blockquantisierer* und *Inverser Blockquantisierer* verschiedene Quantisierungsmatrizen per Parameter auswählen. Eine Übersicht aller wählbaren Matrizen und eine kurze Beschreibung finden Sie in der *Simulink*-Blockübersicht im Kapitel A.[5]

Stellen Sie sicher, dass auf der Empfängerseite die gleichen Blockparameter im Block *Inverser Blockquantisierer* wie im Sender eingestellt sind, damit die Skalierung, die im Encoder durch die Quantisierungsmatrix durchgeführt wird, wieder korrekt rückgängig gemacht wird und das Empfangsbild wie erwartet dargestellt wird.

Stellen Sie der Reihe nach jede Matrix von `1` bis `8` per Parameter einmal ein und lassen jeweils die Simulation mit dieser Matrix durchlaufen. Verwenden Sie als Vorfaktor (also als zweiten Parameter) jeweils `1`! Notieren Sie Ihre Beobachtungen! Deuten Sie die beobachteten Effekte!

> **Hinweis:** Denken Sie daran, die Quantisierungsmatrixnummer und den Vorfaktor (in diesem Versuchsteil konstant `1`!) **jeweils** sowohl dem Block *Blockquantisierer* wie auch dem Block *Inverser Blockquantisierer* korrekt als Parameter zu übergeben!

> **Tipp: Bei Fehlfunktionen oder unerwarteten Ergebnissen** überprüfen Sie, ob Sie die Makroblockgrößen in den Blöcken *Block-DCT* und *Inv. Block-DCT* korrekt eingestellt haben!

> **Hinweis:** Achten Sie darauf, dass im Messkanalmodell der Wertebereich zur Ermittlung der Blockentropien korrekt eingestellt ist! Da Transformationskoeffizienten übermittelt werden sollen, müssen Sie -4096 bis $+4096$ einstellen!

2.) **Was bedeuten die Zahlen in den Quantisierungsmatrizen?**[6] Sie können die Namen der Matrizen als Hinweise verwenden. **Bedeuten große Werte, dass der entsprechende Frequenzanteil stark quantisiert wird oder nicht?**

3.) **Welche Frequenzanteile sind besonders wichtig für die Bildqualität, hochfrequente oder niederfrequente?**

[5] Die Matrizen sind in der Datei *quant_matrix.m* im Ordner **simulink** hinterlegt. Um diese zu betrachten, können Sie diese beispielsweise mit dem MATLAB-Editor öffnen. Achten Sie darauf, dass Sie den Inhalt der Datei nicht ohne vorherige Absprache mit Ihrer Betreuerin oder Ihrem Betreuer verändern!

[6] Siehe Skript S. 20.

4.) Sie haben im Theorieteil gelernt, dass das PSNR als ein Indikator für die Ähnlichkeit zweier Bilder zueinander ist. Damit kann der PSNR-Wert als Gütekriterium für die Transformationscodierung herangezogen werden.
In Ihrem Modell sollte bereits der Block *PSNR berechnen* vorhanden sein. Mithilfe des Blocks *PSNR anzeigen* können Sie die entsprechenden Werte ablesen. Ermitteln Sie nun die PSNR-Werte der ersten fünf Bilder der Videosequenz und bilden Sie den arithmetischen Mittelwert Ihrer Messwerte! (Als Quantisierungsmatrix soll hier die JPEG-Matrix eingestellt sein.)

5.) Wie Ihnen bereits aus dem Theorieteil bekannt ist, ist die letzte Quantisierungsmatrix das Ergebnis umfangreicher Tests und wird in realen Videocodiersystemen verwendet, da sie einen guten Kompromiss zwischen Datenreduktion und guter subjektiver Qualität darstellt. **Betrachten Sie ein beliebiges Bild (zum Beispiel das letzte Ihrer Simulation), welches mit dieser Matrix quantisiert wurde und beurteilen Sie kritisch die Bildqualität. Fallen Ihnen Bildfehler auf? Welcher Art sind diese und wo sowie in welchem Bild (Bildnummer) sehen Sie diese?**

14.6. Dritte Aufgabe: Zeitliche DPCM-Encoder-Implementierung mit Bewegungsschätzung in Simulink

Zeitvorschlag: 40 min

In diesem Versuch arbeiten Sie mit einem einfachen DPCM-**En**coder in *Simulink*. Der Encoder enthält eine Bewegungskompensation (und dadurch einen unverzichtbaren Bewegungsschätzer). Auf die Transformationscodierung wird in diesem Versuch jedoch bewusst verzichtet. Über den Messkanal werden lediglich die geschätzten Bewegungsvektoren sowie das Differenzbild zwischen Schätzung und Originalbild übertragen. Als *Bildspeicher* wird das Verzögerungsglied *Unit Delay* verwendet, welches mit dem Symbol $\frac{1}{z}$ kenntlich gemacht ist und das anliegende Bild um einen Takt verzögert.

Ein Messkanalmodell, den eben beschriebenen Encoder und einen vorbereiteten Decoder auf der Empfängerseite finden Sie in der Datei *dpcm_ohne_frequenztrafo_student*, die Sie zu Beginn des Versuches laden sollten.

Bevor Sie mit dem Versuch beginnen können bedarf es einer Fehleranalyse des Systems. Es hat sich ein Fehler eingeschlichen und nun wird das Bild nicht korrekt prädiziert, Sender und Empfänger laufen auseinander. Gehen Sie das System durch und ergründen Sie, woran die zeitliche Vorhersage scheitert. Dieses Vorgehen wird Ihnen helfen, Ihr System und die Funktionsweise des zeitlichen DPCM besser zu verstehen. Modifizieren Sie das System entsprechend, bis es die korrekte Funktion erfüllt. Als Hilfestellung können Sie die Darstellung in Abbildung 14.3 verwenden. Sie finden alle benötigten Bauelemente, die Sie für diese Aufgabe benötigen, wie gewohnt in der *Videocodierungs-Bibliothek*.

Hinweis: Stellen Sie sicher, dass Sie im *Bewegungsschaetzer*-Block die vorgeschlagene Makroblockgröße von 16 verwenden. Diese ist ebenfalls im Block *Bewegungskompensation* voreingestellt, welcher sich im Subsystem *zeitlicher Praediktor* befindet.
Der Suchtiefeparameter p (zweiter Parameter im *Bewegungsschaetzer*-Block) muss 7 sein!

Hinweis: Nach der Durchführung rot markierte Blöcke und Verbindungen weisen NICHT auf den Fehler hin.

Tipp: Zur Visualisierung des Differenzsignals sollten Sie den Wertebereich des *Zeige Bild in Fenster <x>*-Blocks reduzieren, damit Sie auch geringe Differenzen erkennen können. Wählen Sie als Grenzen etwa −30 und +30!

Als Hilfestellung können Sie die Darstellung in Abbildung 14.2 verwenden.

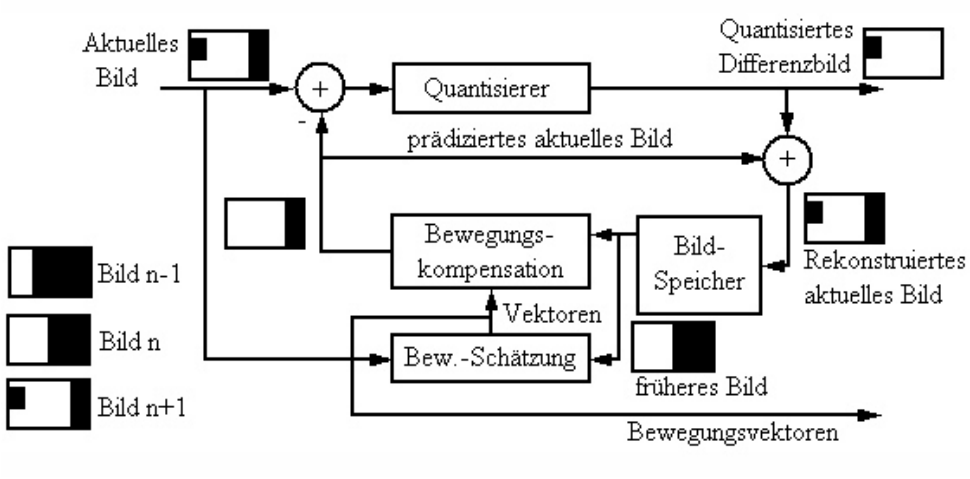


Abbildung 14.2.: DPCM-Prädiktor (aus [7])

Wenn Sie alle Elemente korrekt angeordnet haben, sollte die Simulation ohne Fehlermeldung funktionieren. Falls eine Fehlermeldung auftritt, lesen Sie diese aufmerksam durch. Zwar sehen die Fehlermeldungen auf den ersten Blick kryptisch aus, jedoch enthalten sie zumeist Hinweise, welcher Block in welcher Codezeile den Fehler verursacht und Sie müssen in den Blöcken auf bekannte Art und Weise die korrekten Parameter einstellen, die für die Ausführung benötigt werden. Sollten Sie Schwierigkeiten mit der Programmumgebung haben, kann Ihr Betreuer Ihnen beim Debuggen und bei der korrekten Einstellung der Parameter behilflich sein.

Für empfohlene Parametereinstellungen ziehen Sie eventuell erneut das Kapitel A zu Rate.

Fragen zum zeitlichen DPCM-System

1.) **Stellen Sie im Quantisierer-Block die Quantisierung auf 1 (verlustlos), und lassen Sie Ihre Simulation laufen. Notieren Sie sowohl die Entropien wie auch die *Datenwerte ungleich null* für die ersten fünf Bilder.** Es steht eine Tabelle im Protokollkapitel 15.3 für Sie bereit.

Hinweis: Auch hier können Sie sich wieder die Ausgaben im *Command Window* zu Nutze machen.

2.) **Wie haben sich die *Entropien* verglichen mit den Entropien, die Sie im Versuch zur „örtlichen Prädiktion“ aus Kapitel 14.4 beobachtetet haben, qualitativ verändert? Sind sie größenordnungsmäßig im Mittel gestiegen, gefallen oder gleich geblieben? Was ist grundlegend anders, sprich, was für eine Prädiktion ist hier eingesetzt worden?** (Es genügt eine stichwortartige Antwort auf diese Frage!)

3.) **Warum ist die Entropie des ersten Bildes *systembedingt* höher als die Entropie folgender Bilder?**

4.) **Warum sind trotz der DPCM mit Prädiktionscodierung immer noch sehr viele *Datenwerte ungleich null*?**

5.) Zwar haben Sie eine Veränderung in der Entropie beobachtet, jedoch müssen Sie nach wie vor ein komplettes Bild vom Empfänger zum Sender befördern. Auf den ersten Blick ist also keine Dateneinsparung zu erkennen, im Gegenteil scheinen sogar in Form der Bewegungsvektoren, die ebenfalls über den Messkanal übertragen werden müssen, *mehr* Daten als bisher anzufallen.

Wie können Sie nun den beobachteten Effekt dennoch gewinnbringend in der Praxis nutzen? Welche Maßnahmen müssen Sie dazu ergreifen und warum würden sie etwas nützen, d. h. wie funktionieren diese?

14.7. Vierte Aufgabe: Vollständige Hybridcodierung

Zeitvorschlag: 20 min

Für den letzten Versuch haben wir die DPCM-Encoder-Decoder-Kette aus Versuch 14.6 und die Transformationscodierung aus Versuch 14.5 zu einem hybriden System zusammengefasst. Das System befindet sich in der Datei *hybridcodierung_vollstaendig_ohne_psnr_student*. Stellen Sie sicher, dass für die erste Aufgaben im Block *Blockquantisierer* bzw. *inverser Blockquantisierer* die Quantisierungsmatrix `quant` bzw. `8` eingestellt ist.

Fragen zur vollständigen Hybridcodierung

1.) **Wie hat sich nun die Entropie im Vergleich zum zeitlichen DPCM-System *ohne* Transformationscodierung verändert?**

2.) **Beobachten Sie erneut die Anzeige der *Datenwerte ungleich 0*. Wie haben sich die Werte qualitativ im Vergleich zum Vorversuch *ohne* Transformation verändert? Sind sie gestiegen oder gesunken? Interpretieren Sie die Ergebnisse und geben Sie an, welche Elemente des Übertragungssystems für das beobachtetete Verhalten hauptsächlich verantwortlich sind!**

3.) Als letztes sollen Sie die Leistungsfähigkeit in Bezug auf die Bildqualität des hybriden Codiersystems erleben. **Schließen Sie an Ihr Hybridsystem in der Encodereinheit die beiden Blöcke *PSNR berechnen* und *PSNR-Anzeige* aus der *Videocodierungsbibliothek* an. Dem *PSNR berechnen*-Block müssen Sie zum einen als Referenzbild das Originalbild vor der Quantisierung, zum anderen das Empfangsbild zuführen. Stellen Sie die Matrix `quant` im *Blockquantisierer* ein, sofern diese nicht ohnehin aus dem Vorversuch noch eingestellt ist und messen Sie das Spitzen-Signal-Rauschverhältnis PSNR der ersten fünf Bilder der Videosequenz *susie.yuv*.**

4.) **Vergleichen Sie die eben ermittelten PSNR-Werte und Entropien mit denen aus der reinen Transformationscodierung (Kapitel 14.5)! Interpretieren Sie das Ergebnis!**

5.) Sie können den *Simulink*-Blöcken *Blockquantisierer* und *Inverer Blockquantisierer* jeweils einen Vorfaktor für die gewählte Quantisierungsmatrix als zweiten Parameter übergeben. **Ermitteln Sie durch Probieren verschiedener Multiplikatoren den Faktor, den Sie benötigen, um im Mittel ein PSNR $\geq 40\,\mathrm{dB}$ für die ersten fünf Bilder garantieren zu können!** (Die Angabe des Ergebnisses auf eine Nachkommastelle genau genügt!)

> **Hinweis:** Denken Sie daran, die Quantisierungsmatrixnummer und den gewählten Vorfaktor **jeweils** allen *Blockquantisierer*- und *Inverser Blockquantisierer*-Blöcken korrekt als Parameter zu übergeben! Verwenden Sie als Dezimalzeichen den Punkt ».«!

14.8. Zusammenfassung

Sie haben nun ein Videokompressionssystem geschaffen, welches modernen Systemen im wesentlichen entspricht. Natürlich haben Sie in diesem Versuch vereinfachte Elemente kennengelernt. Beispielsweise kann alleine für die Bewegungsschätzung nahezu beliebig viel Aufwand getrieben werden um ein System zu optimieren. Zusätzlich kommen vielerlei weitere Techniken zum Einsatz, wie etwa ein (verbessertes) Deblocking Filter, Zickzack-Abtastung, Erhöhung der Prädiktionstiefe[1] etc..
Im Kern jedoch bestehen alle heute im Einsatz befindlichen Systeme auch modernster Bauart aus einer Verbindung (*hybrid*!) der Elemente, die Sie heute kennen gelernt haben und mit denen Sie sich im praktischen Teil dieses Versuchs beschäftigt haben, nämlich aus:

- **örtlicher Prädiktion**
- **zeitlicher Prädiktion mit Hilfe von DPCM**
- **Entropiecodierung (VLC[2] z. B. Huffman-Codierung oder arithmetische Codierung zur Ausnutzung der verringerten Entropie der Differenzbilder aus den ersten beiden Punkten)**
- **Transformationscodierung**

[1] d. h., dass beliebige Vorbilder zur Prädiktion herangezogen werden können
[2] *VLC* steht für *variable length coding*.

15. Protokolle

15.1. Versuchsprotokoll zur örtlichen Prädiktion

zu 1.) **Für das mit 8 bit quantisierte Quellmaterial erwarte ich eine Entropie von zirka (Einheit mit angeben!):**

Die maximale Entropie beträgt (Einheit mit angeben!):

zu 2.) **Die gemessene Entropie [in bit] beträgt ungefähr:**

zu 3.) **Was wird in den sich öffnenden Fenstern dargestellt?**

zu 4.) **Was passiert, wenn Sie im senderseitigen Prädiktor eine andere Prädiktionsmatrix vorgeben als empfängerseitig?**

zu 5.)

- **8 bit quantisiertes Quellmaterial *ohne* Prädiktionscodierung:**

- **8 bit quantisiertes Quellmaterial *mit* Prädiktionscodierung:**

15.2. Versuchsprotokoll zur Transformationscodierung

zu 1.)

Matrixname	Beobachtung	Deutung / Erklärung
quant_no		
quant0		

Fortsetzung auf nächster Seite

Matrixname	Beobachtung	Deutung / Erklärung
quant00		
quant0x		
quantx0		
quanthigh		
quantlow		
quant		

Tabelle 15.1.: Versuchsprotokoll zur Transformationscodierung

zu 2.) **Was bedeuten die Zahlen in den Quantisierungsmatrizen?**

zu 3.) **Welche Frequenzanteile sind besonders wichtig für die Bildqualität, hochfrequente oder niederfrequente?**

zu 4.)

Bildnummer	PSNR in dB
1	
2	
3	
4	
5	
Mittelwert:	

Tabelle 15.2.: Versuchsprotokoll zur Transformationscodierung: PSNR-Werte

zu 5.) **Fallen Ihnen Bildfehler auf? Welcher Art sind diese und wo sowie in welchem Bild (Bildnummer) sehen Sie diese?**

15.3. Versuchsprotokoll zum DPCM-System

zu 1.)

Bildnummer	Entropie	Datenwerte != 0
1		
2		
3		
4		
5		

Tabelle 15.3.: Versuchsprotokoll zum DPCM-System

zu 2.) **Wie haben sich die *Entropien* verglichen mit den Entropien, die Sie im Versuch zur „örtlichen Prädiktion“ aus Kapitel 14.4 beobachtetet haben, qualitativ verändert?**

zu 3.) **Warum ist die Entropie des ersten Bildes *systembedingt* drastisch höher als die Entropie folgender Bilder?**

zu 4.) **Warum sind trotz der DPCM mit Prädiktionscodierung immer noch sehr viele *Datenwerte ungleich null*?**

zu 5.) **Wie können Sie nun den beobachteten Effekt dennoch gewinnbringend in der Praxis nutzen? Welche Maßnahmen müssen Sie dazu ergreifen und warum würden sie etwas nützen, d. h. wie funktionieren diese?**

15.4. Versuchsprotokoll zum vollständigen hybriden Videocodierungssystem

zu 1.) **Wie hat sich nun die Entropie im Vergleich zum zeitlichen DPCM-System *ohne* Transformationscodierung verändert?**

zu 2.) **Beobachten Sie erneut die Anzeige der *Datenwerte ungleich 0*. Wie haben sich die Werte qualitativ im Vergleich zum Vorversuch *ohne* Transformation verändert? Sind sie gestiegen oder gesunken? Interpretieren Sie die Ergebnisse und geben Sie an, welche Elemente des Übertragungssystems für das beobachtetete Verhalten hauptsächlich verantwortlich sind!**

zu 3.)

Bildnummer	PSNR in dB
1	
2	
3	
4	
5	

Tabelle 15.4.: PSNR-Messung im vollständigen Hybridsystem

zu 4.) **Vergleichen Sie die eben ermittelten PSNR-Werte und Entropien mit denen aus der reinen Transformationscodierung (Kapitel 14.5)! Interpretieren Sie das Ergebnis!**

zu 5.) **Benötigter Vorfaktor für ein PSNR $\geq 40\,\mathrm{dB}$:**

Anhang

A. Simulink Blöcke und Parameter

Nachfolgend sind alle Funktionen und Parameter der Simulationsblöcke aufgeführt, die für die Praktikumsaufgaben benötigt werden. Parameterwerte sind in das Fenster einzutragen, welches nach einem Doppelklick auf einen Block erscheint. Mehrere Parameter sind durch Kommata »,« zu trennen. Als Dezimaltrennzeichen muss der Punkt ».« verwendet werden, Datei- und Pfadangaben müssen in Hochkomma »'« gesetzt werden. Die Pfadangabe sollte betriebssystemunabhängig der Unix- / Linuxkonvention folgend ein »/« zur Trennung von Ordnerebenen enthalten.

Die Blöcke sind nach erstmaligem Erscheinen innerhalb der Versuchsteile und nicht nach Alphabet sortiert.

Videodatei anschauen

Dieser Block spielt YUV-Rohdatenvideomaterial mittels der MATLAB-Funktion `movie` ab. Die Parameter und Mustereinstellungen können folgender Tabelle entnommen werden:

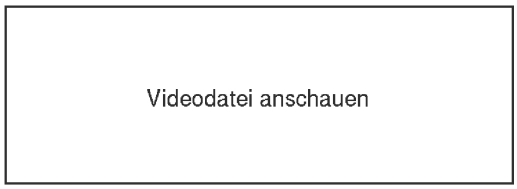


Abbildung A.1.: *Videodatei anschauen-Simulink*-Block

Blockparameter	**Mustereinstellung**
YUV-Videodatei in Hochkomma »'«	`'../videos/susie.yuv'`
Videobreite in Pixeln	`352`
Videohöhe in Pixeln	`288`
Nummer des ersten abzuspielenden Frames	`1`
Nummer des letzten abzuspielenden Frames	`89`
Framedatenrate [in $\frac{\text{Frames}}{\text{Sekunde}}$]	`25`
Anzahl der Abspielwiederholungen	`2`
Nummer des Videofensters	`3`
vertikale Position des Videofensters[1]	`150`
horizontale Position des Videofensters[1]	`150`

Bildtaktgeber

Vorliegendes Element regelt, welches Bild aus einer *Videodatei* gelesen werden soll. Der Laufbereich ist einstellbar unter `Simulation` → `Configuration Parameters` (bzw. Druck auf STRG + E). Es dient als Eingangssignal für den *Simulink*-Block *Lese Bild aus Datei (Bildquelle)* und steuert, welches Bild dieser aus dem Videostream lesen soll.

Abbildung A.2.: *Bildtaktgeber-Simulink*-Block

Blockparameter	**Mustereinstellung**
keine	

[1] **Hinweis:** In MATLAB beginnt die Zählung der Bildschirmkoordinaten am unteren linken Bildschirmrand mit $(0, 0)$. Die erste Koordinate kennzeichnet die horizontale Richtung, die zweite die vertikale.

Lese Bild aus Datei (Bildquelle)

Dieser Block liest aus einer Videoquelldatei im YUV-Rohdatenformat einzelne Bilder aus. Am Eingang muss ein Bildtaktgeber angeschlossen werden, der regelt, welches Bild gelesen werden soll. An den Ausgängen wird das Video in Einzelbildern ausgegeben. Es ist zerlegt in die Luminanz Y und die beiden Chrominanzen U und V. Als Parameter müssen der Name der Videoquelldatei inklusive Pfadangabe sowie die Pixelanzahl in horizontaler und vertikaler Richtung angegeben werden.

Abbildung A.3.: *Lese Bild aus Datei-Simulink*-Block

Beispielparameter sind folgender Auflistung zu entnehmen:

Blockparameter	**Mustereinstellung**
YUV-Videodatei in Hochkomma »'«	`'../videos/susie.yuv'`
Videobreite in Pixeln[2]	352
Videohöhe in Pixeln[2]	288

Zeige Bild in Fenster <x>

Dieser Block zeigt das ihm am Eingang übergebene Bild im Fenster mit der Nummer <x>, welches in den Parametern nebst Position eingestellt werden kann, an. Die Parameter haben dabei folgende Bedeutung:

Abbildung A.4.: *Zeige Bild in Fenster <x>-Simulink*-Block

Blockparameter	**Mustereinstellung**
Fensternummer	10
untere Grenze des darzustellenden Wertebereichs (für negative Eingaben wird die Anzeige in den positiven Bereich verschoben und danach auf einen 8 bit Wertebereich normiert	0
obere Grenze des darzustellenden Wertebereichs	255
Position der unteren linken Fensterecke in x-Richtung[2]	10
Position der unteren linken Fensterecke in y-Richtung[2]	380

[2] **Hinweis:** In MATLAB beginnt die Zählung der Bildschirmkoordinaten am unteren linken Bildschirmrand mit $(0, 0)$. Die erste Koordinate kennzeichnet die horizontale Richtung, die zweite die vertikale.

Histogramm in neuem Fenster anzeigen

Durch den Block wird ein neues Fenster kreiert, in welchem ein Histogramm erstellt wird, in dem die Grauwertverteilung des aktuellen Bildes zu sehen ist.

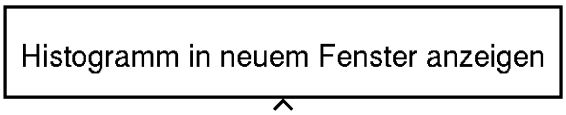
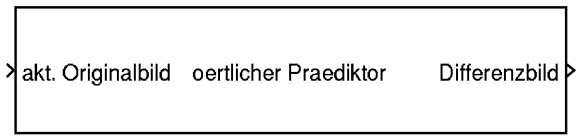


Abbildung A.5.: *Histogramm in neuem Fenster anzeigen-Simulink*-Block

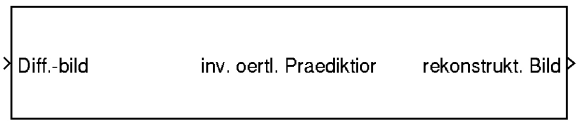
Blockparameter	**Mustereinstellung**
Fensternummer	10
Position der unteren linken Fensterecke in x-Richtung[2]	10
Position der unteren linken Fensterecke in y-Richtung[2]	380

Örtlicher Prädiktor / Inverser örtlicher Prädiktor

Technisch gesehen sind beide hier beschriebenen Blöcke gleich. Sie führen eine örtliche Prädiktion anhand einer übergebenen Prädiktionsmatrix (siehe Kapitel 6.1) für das Bild, welches am Eingang anliegt, durch und geben das Ergebnis als Matrix zurück.
Jedem Block kann über eine entsprechende Einstellung eine eigene Prädiktionsmatrix übergeben werden, so dass auch der Effekt verschiedener Matrizen beim Encodier- und Decodierprozeß simulierbar ist. Die übergebene Prädiktionsmatrix enthält die Gewichtungskoeffizienten der Nachbarpixel, wobei das zweite Element der Zahlengruppe nach dem Semikolon dem aktuellen Pixel entspricht und das Element links davon dem Bildpunkt links davon, also dem Pixel (-1,0). Die erste Zahlengruppe bezeichnet die Pixel, die oberhalb des aktuellen Bildpunktes liegen, also die Pixel (-1,-1), (0,-1) und (1,-1) Eine graphische Darstellung findet sich im Theorieteil in Abbildung 6.2.
Im Versuch dürfen nur die Pixel (0,-1) und (-1,0) ungleich null sein.



(a) *oertlicher Praediktor-Simulink*-Block



(b) *inverser oertlicher Praediktor-Simulink*-Block

Abbildung A.6.: *Örtliche Prädiktoren*

Blockparameter	**Mustereinstellung**
Prädiktionsmatrix	`[0 0.5 0; 0.5 0 0]`

Messkanal

Der *Messkanal* oder das *Messkanalmodell* ist ein zentraler Bestandteil eines jeden Versuchsteils. Im engeren Sinne ist mit *Kanal* der Übertragungsweg zwischen Sender und Empfänger gemeint. Das kann sowohl eine Freiluftübertragung, eine Satellitenverbindung, aber auch ein optisches Medium wie eine DVD oder Bluray-Disc sein – eben alles, was sich zwischen einer datensendenden und einer datenempfangenen Einheit befindet.

Da aber **alle** Daten aufgrund eben geschilderter Beschaffenheit des Kanals jenen passieren müssen, bietet er eine hervorragende Visualisierung eines physikalischen Kanals und eignet sich zudem zur Ermittlung bestimmter Kenngrößen wie der Entropie der Messkanaldaten, aus denen wiederum Rückschlüsse auf die benötigte Datenrate gezogen werden können. Daher heißt der Kanal, der zudem Messungen durchführen kann, in den Simulinkmodellen *Messkanal*. Er nimmt an seinem ersten Eingang ein Bild oder Differenzbild entgegen, welches direkt durchgeleitet wird und am ersten Ausgang direkt wieder abgegriffen werden kann. An den zweiten Eingang sollen – sofern vorhanden – Bewegungsvektoren eingekoppelt werden, die ebenfalls direkt auf der Empfängerseite am entsprechend gekennzeichneten Ausgang zur Verfügung stehen.

Darüber hinaus stehen weitere Ausgänge zur Verfügung, die je nach Versuchsteil und Aufgabenstellung beschaltet oder einfach ignoriert werden können. An die Ausgänge, die mit *Entropie*, *Blockentropie*, *mittlere Blockentropie* und *Datenwerte != 0* beschriftet sind, können *Display*-Einheiten angeschlossen werden um die vom *Messkanal*-Block ermittelten Werte anzuzeigen.

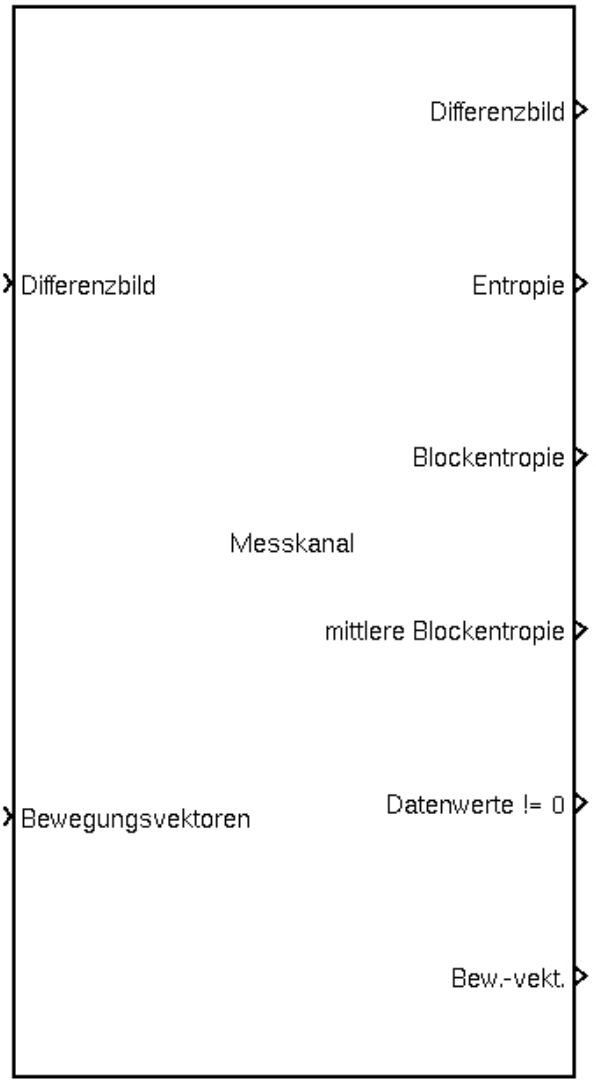


Abbildung A.7.: *Messkanalmodell-Simulink*-Block

Der Einheitlichkeit der Praxisanteile wegen existiert im gesamten Versuch nur *ein* Messkanalmodell. Der Nachteil ist zwar, dass je nach Aufgabenstellung einige Ein- und Ausgänge ungenutzt bleiben, der große Vorteil – der diesen Nachteil jedoch aufwiegt – besteht darin, dass auch anschaulich *der „Kanal“ immer der gleiche* bleibt, egal welche Daten darüber übertragen werden. Es stehen drei Paramter zur Verfügung, deren Bedeutung nachfolgend tabelliert sind:

Blockparameter	Mustereinstellung
Blockgröße zur Berechnung der Blockentropie	`8`
untere Grenze des Wertebereichs, der zur Berechnung der Blockentropie herangezogen wird (Auswertung mit Matlab-Funktion `hist`, negative Werte zulässig, wenn Transformationsbilder statt Differenzbilder übertragen werden sollen)	`0` oder `-4096`
obere Grenze des Wertebereichs zur Auswertung der Blockentropie	`255` oder `4096`

Entropie

Es handelt sich bei dem *Entropie*-Block um ein gewöhnliches *Display*-Element aus der Simulink Standardbibliothek *Simulink* → *Sink*. Am Eingang muss ein Skalar anliegen, der im Display des Blockes direkt während einer laufenden Simulation angezeigt und ständig aktualisiert wird, sobald sich der Eingangswert ändert. Der Block kann beispielsweise an den entsprechenden *Messkanal*-Ausgang angeschlossen werden. Parameter können keine eingestellt werden.

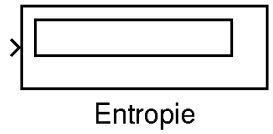


Abbildung A.8.: *Entropieanzeige-Simulink*-Block

Blockparameter	Mustereinstellung
keine	

Block-DCT / Inverse Block-DCT (Block IDCT)

Da beide Blöcke auf der gleichen Codebasis beruhen und ihre Einstellungen somit identisch sind, werden beide Blöcke in einem Absatz beschrieben.

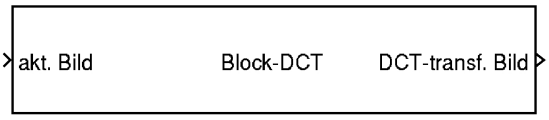


Abbildung A.9.: *Block DCT-Simulink*-Block

Die *Block DCT* bzw. die *Block IDCT* führt eine DCT (Diskrete Kosinustransformation) bzw. eine IDCT (inverse diskrete Kosinustransformation) durch. Zur Bestimmung der Spektraldarstellung werden MATLAB-eigene Funktionen genutzt.

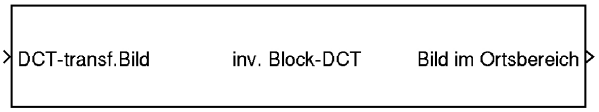


Abbildung A.10.: *inverse Block-DCT-Simulink*-Block

Blockparameter	Mustereinstellung
Kantenlänge eines Blocks	`8`

Blockquantisierer / Inverser Blockquantisierer

Der *Blockquantisierer* führt anhand einer per Parameter ausgewählten Matrix, die mit dem per Parameter wählbaren Vorfaktor multipliziert wird, eine punktweise Division mit Rundung des Eingangsmaterials durch. Es sind bereits einige Matrizen, die für die Versuchsdurchführung benötigt werden, vorbereitet. Die Matrizen selber sind in der Funktion `quant_matrix.m` zu finden, welche sich im **simulink**-Ordner befindet. Sinnvoll ist eine solche Quantisierung in Verbindung mit einer *Block DCT* um bestimmte spektrale Koeffizienten stark zu betonen oder zu unterdrücken.

Abbildung A.11.: *Blockquantisierer-Simulink*-Block

In jedem Falle **müssen** sender- wie empfängerseitig die gleichen Einstellungen verwendet werden, da ansonsten mit einem falsch normierten Bild auf Empfängerseite weitergearbeitet wird.

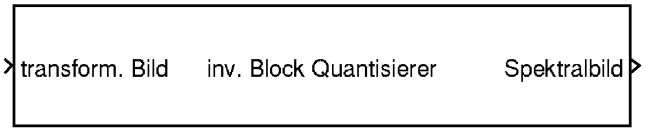


Abbildung A.12.: *inverser Blockquantisierer-Simulink*-Block

Blockparameter	**Mustereinstellung**
Nummer der Quantisierungsmatrix	`1 ...8`
Vorfaktor, mit dem die Quantisierungsmatrix multipliziert wird	`1`

Nummer ...	**Name ...**	**Eigenschaften der Quantisierungstabelle**
1	quant_no	keine Quantisierung, alle Koeffizienten 1
2	quant_0	höchste Quantisierungsstufe, alle Koeffizienten 5000
3	quant_00	alle Koeffizienten der Quantisierungsmatrix 5000 außer an Position (1,1), dort Koeffizient gleich 1
4	quant_0x	erste Spalte der Quantisierungsmatrix 1, sonst alle Koeffizienten 5000
5	quant_x0	erste Zeile der Quantisierungsmatrix 1, sonst alle Koeffizienten 5000
6	quant_high	Koeffizienten an Position (1,1), (1,2), (2,1) 5000, restliche Koeffizienten 1
7	quant_low	Koeffizienten in der linken oberen Matrixhälfte sind 1, alle anderen 5000 (diagonal unterteilt).
8	quant	Quantisierungsmatrix laut Standard für MPEG und JPEG

PSNR berechnen

Der Block *PSNR berechnen* benötigt keine Parameter. Er ermittelt basierend auf den beiden am Eingang anliegenden Bildern das Spitzen-Signal zu Rausch-Verhältnis und stellt dieses als Skalar am Ausgang zur Verfügung. Anzuschließen ist ein *Display*-Element oder das angepaßte Displayelement *PSNR betrachten*.
Werte von 30 – 40 dB signalisieren eine gute bis sehr gute Bildqualität, wenngleich der PSNR-Wert auch kein allgemein gültiges Qualitätsmaß darstellt.

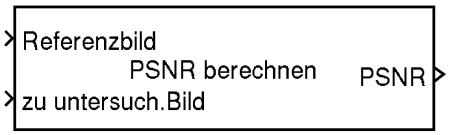


Abbildung A.13.: *PSNR berechnen-Simulink*-Block

Blockparameter	Mustereinstellung
keine	

PSNR betrachten

Es handelt sich bei dem *PSNR anzeigen*-Block um ein gewöhnliches *Display*-Element aus der Simulink Standardbibliothek *Simulink* → *Sink* und wird an den Block *PSNR berechnen* angeschlossen. Am Eingang muss ein Skalar anliegen, der im Display des Blockes direkt während einer laufenden Simulation angezeigt und ständig aktualisiert wird, sobald sich der Eingangswert ändert.

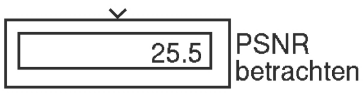


Abbildung A.14.: *PSNR betrachten-Simulink*-Block

Blockparameter	Mustereinstellung
keine	

Anzahl Datenwerte != 0

Es handelt sich bei dem *Anzahl Datenwerte != 0*-Block[3] um ein gewöhnliches *Display*-Element aus der Simulink Standardbibliothek *Simulink* → *Sink*. Es wird an den *Messkanal* an den entsprechend beschrifteten Ausgang angeschlossen und stellt im Display des Blockes direkt zur Laufzeit einer Simulation den am Eingang jeweils anliegenden Skalar dar.

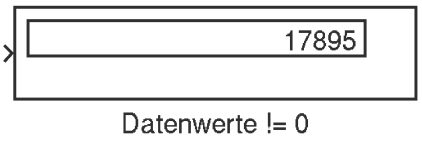


Abbildung A.15.: *Anzahl Datenwerte ungleich null-Simulink*-Block

Blockparameter	Mustereinstellung
keine	

Blockentropien

Es handelt sich bei dem *Blockentropien-Simulink*-Block um ein gewöhnliches *Display*-Element aus der Simulink Standardbibliothek *Simulink* → *Sink*. Es muss eine Matrix am Eingang anliegen, deren Werte in einzelnen Feldern des Blockes direkt während einer laufenden Simulation angezeigt und bei sich verändernen Werten ständig aktualisiert werden. Der Block wird an den Ausgang des *Messkanals* angeschlossen.

Die Blockentropien repräsentieren die Entropien der Transformationskoeffizienten der DCT. Somit stellt der linke obere Wert die Entropie des Gleichanteils eines Bildes dar, der rechte untere die Entropie des hochfrequentesten Anteils im Bild.

[3] Der Ausdruck »!=« steht für *ungleich*.

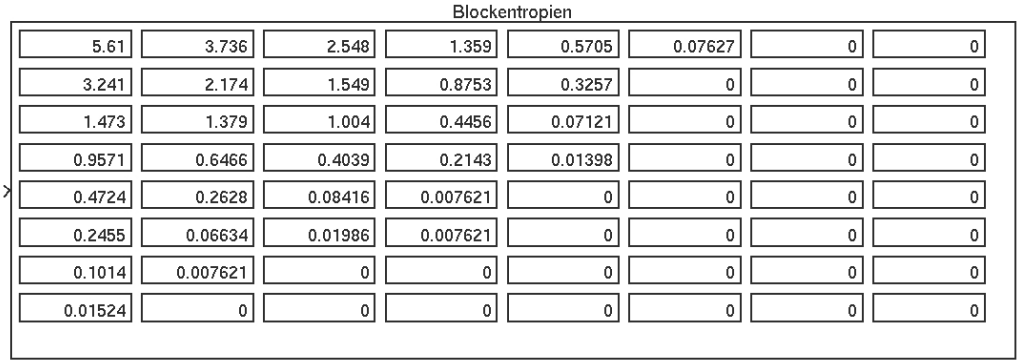


Abbildung A.16.: *Blockentropien-Simulink*-Block

Blockparameter	**Mustereinstellung**
keine	

Mittlere Blockentropie

Es handelt sich bei diesem Block um ein gewöhnliches *Display*-Element aus der Simulink Standardbibliothek *Simulink* → *Sink*, welches den am Eingang anliegenden Skalar zur Laufzeit einer Simulation anzeigt. Es wird an den Messkanal an den entsprechenden Ausgang angeschlossen und stellt das arithmetische Mittel aller Einzelentropien des Blockes *Blockentropien* dar.

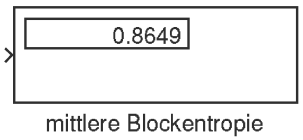


Abbildung A.17.: *mittlere Blockentropie-Simulink*-Block

Blockparameter	**Mustereinstellung**
keine	

Zeitlicher Praediktor

Dieser Block repräsentiert ein Subsystem in Simulink, welches durch Doppelklick auf das Blockschaltbildsymbol geöffnet werden kann. Einstellungen innerhalb des Subsystems sind für die Versuchsdurchführung nicht notwendig, der Praediktor darf als *Black Box* angesehen werden, in welche Bewegungsvektoren und ein Referenzbild hineingesteckt werden, woraufhin sie ein aktuelles Bild *vorhersagt*.

Abbildung A.18.: *zeitlicher Prädiktor-Simulink*-Block

Blockparameter	**Mustereinstellung**
keine	

Quantisierung

Dieser Block Quantisiert das Eingangsbild mit der als Parameter übergebenen Quantisierungsschrittweite. Eine Quantisierungsschrittweite von 1 lässt das Bild unangetastet.

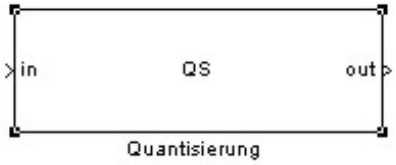


Abbildung A.19.: *Quantisierung-Simulink*-Block

Blockparameter	**Mustereinstellung**
Quantisierungsschrittweite	6

"Bildspeicher" = 1 Bild Verzoegerung

Es handelt sich bei diesem Block um einen gewöhnlichen *Unit Delay*-Block ($\frac{1}{z}$) aus der Simulink Standardbibliothek *Simulink* → *Discrete*. Der Block wird als Simulation eines Bildspeichers, welcher **ein** Bild zwischenspeichert, eingesetzt.

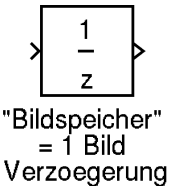


Abbildung A.20.: *Bildspeicher-Simulink*-Block

Blockparameter	**Mustereinstellung**
keine	

Bewegungskompensation

Dieser Block ist Bestandteil und Kernstück der *Black Box „Zeitlicher Praediktor“* (siehe oben). Er ist nur der Vollständigkeit halber gelistet. Er schätzt aus dem Bild am ersten Eingang unter Verwendung der Bewegungsvektoren, die am zweiten Eingang anliegen, ein bewegungskompensiertes Bild.

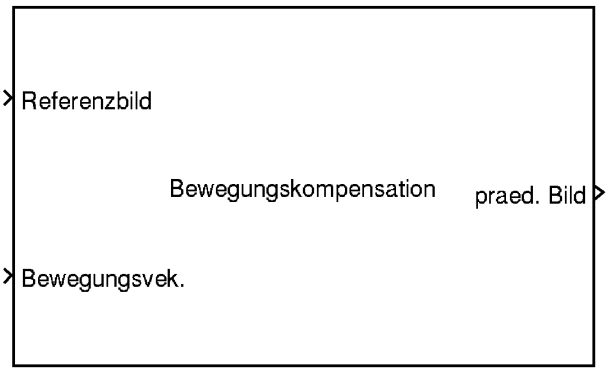


Abbildung A.21.: *Bewegungskompensation-Simulink*-Block

Blockparameter	**Mustereinstellung**
Kantenlänge eines Makroblocks in Pixeln	16

Bewegungsschaetzer

Der *Bewegungsschätzer* schätzt aus einem Referenzbild und dem aktuellen Bild Bewegungsvektoren. Dabei wird eine *Diamondsearch* zur Ermittlung der Bewegung verwendet. Blockweise wird so für jeden Makroblock ein Bewegungsvektor ermittelt, aus dem unter Zuhilfename des für die Prädikton verwendeten Referenzbildes letztlich das aktuelle Bild geschätzt werden kann (siehe Block *Bewegungskompensation*).

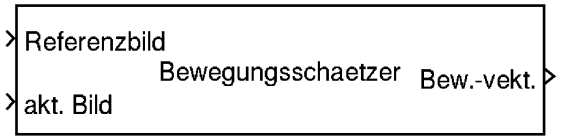


Abbildung A.22.: *Bewegungsschätzer-Simulink*-Block

Blockparameter	**Mustereinstellung**
Kantenlänge eines Makroblocks in Pixeln	16
Parameter »p« für die Suchtiefe	7

Sum / Diff

Es handelt sich bei diesem Block um einen gewöhnlichen *Summierer*-Block aus der Simulink Standardbibliothek *Simulink* → *Math Operations*.

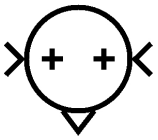


Abbildung A.23.: *Summierer-Simulink*-Block

Über die Parameter können Summen und Differenzen über beliebig viele Eingänge definiert werden durch Ergänzen von + oder −.

Der Ausgang wird durch einen senkrechten Strich („|“) in den Parametern (die in diesem Fall nicht durch Kommata getrennt werden müssen) gekennzeichnet. Die Parameteranzahl kann zwischen 0 und ∞ variieren, je nachdem wieviele Eingänge benötigt werden.

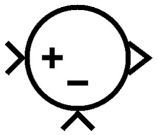


Abbildung A.24.: *Differenzelement-Simulink*-Block

Blockparameter	**Mustereinstellung**
siehe Text, in den Versuchen keine weiteren benötigt	

Literaturverzeichnis

[1] *Duden, Band 5: Das Fremdwörterbuch.* Bibliographisches Institut & F.A. Brockhaus Setzerei GmbH, 7. Auflage, 2001.

[2] Eden, Arnd: *H.264 - Systemanalyse und Evaluation der Leistungsfähigkeit des Codecs*, 2004.

[3] Janssen, Jan Keno: *Philips gibt seine 3D-Sparte auf.* c't - magazin für computer technik, Ausgabe 09'2009 vom 14.04.2009, 2009.

[4] Kürner, Thomas: *Foliensammlung zur Vorlesung Codierungstheorie am Institut für Nachrichtentechnik der Technischen Universität Braunschweig*, 2007. Stand WS 2007/2008.

[5] Kürner, Thomas: *Foliensammlung zur Vorlesung Modellierung und Simulation von Mobilfunksystemen am Institut für Nachrichtentechnik der Technischen Universität Braunschweig*, 2008. Stand Sommersemester 2008.

[6] Ohm, Jens Rainer: *Digitale Bildcodierung.* Springer-Verlag, 1995.

[7] Reimers, Ulrich: *Bildkommunikation II – Sammlung von Arbeitsblättern*, 2017. Stand: Frühjahr 2017.

[8] Strutz, Tilo: *Bilddatenkompression.* vieweg-Verlag, 3. aktualisierte und erweiterte auflage Auflage, 2005.

Abbildungsverzeichnis

Tabellenverzeichnis