Datenkompression

Theorie

1 Einleitung

Übersicht

Wie lassen sich beliebige Daten verdichtet (komprimiert) darstellen?

- verlustfreie Kompression:
  - –
  - –
- verlustbehaftete Kompression:
  - –
  - –

Kompressionsfaktor:

Achtung: Der Kehrwert des Kompressionfaktors wird *Kompressionsrate* genannt. Ein Kompressionsfaktor von 0.5 entspricht der Kompressionsrate 2.

2 Lauflängencodierung

Das Prinzip

Englische Bezeichnung: Run Length Encoding = RLE

- Wähle ein (Steuer-)Zeichen, das in den Daten gar nicht oder selten vorkommt.
- Kommt ein Zeichen in den Daten mehrfach nacheinander vor, wird diese Sequenz durch drei Zeichen ersetzt:
  - Das erste Zeichen ist das mehrfach auftretende Zeichen.
  - Das zweite Zeichen ist das Steuerzeichen.
  - Das dritte Zeichen codiert die Häufigkeit des ersten Zeichens.
- Kommt das Steuerzeichen selbst in den Daten vor, wird ihm das Zeichen für die Null nachgestellt.
- Es können jeweils nur so viele Zeichen zusammengefasst werden, wie es das Zeichen für die Häfigkeitscodierung erlaubt.

Beispiele

Steuerzeichen: das Rautensymbol `#`

Zählzeichen: 4, 5, 6, 7, 8, 9

1. `AAAAA`
2. `ABBCCCDDDD`
3. `AAAAA#BBBB`
4. `XXXXXXXXXX`

Anwendung

-
-

3 Huffman-Codierung

Das Prinzip

Zeichen, die häufiger vorkommen, werden mit weniger Bits codiert als Zeichen, die seltener vorkommen. Dadurch erhalten wir Codes mit variabler Bitlänge.

Um einen solchen Code wieder decodieren zu können, darf kein Codewort den Beginn eines anderen Codes darstellen. (präfixfreier Code)

Sind die folgenden Codes präfixfrei?

- {`1, 01, 001, 0001`}
- {`10, 01, 001, 101`}

Die Codierung

- Bestimme für jedes Zeichen z_i seine Häufigkeit h_i.
- Füge die Paare (h_i, z_i) [=Blätter] nach h_i aufsteigend sortiert in eine Liste [=Wald] ein.
- Führe den folgenden Schritt so oft durch, bis der Wald nur noch aus einem einzigen Baum besteht (Codetabelle).
  - Fasse die ersten beiden Paare (h_1, z_1) und (h_2, z_2) zu einem Baum zusammen, wobei im Wurzelknoten die Summe der Häufigkeiten $h_1 + h_2$ steht. Sortiere diesen Baum anhand seiner Häufigkeit in den Wald ein.
- Um das Codewort eines Zeichens zu bestimmen, durchläuft man den (binären) Baum vom Wurzelknoten aus. Im Wurzelknoten besteht das Codewort aus der leeren Zeichenkette. Für jeden Abzweigung nach links, hängt man eine `0` an das Codewort an. Für jeden Abzweigung nach rechts, hängt man eine `1` an das Codewort an.

Die Decodierung

Voraussetzung: Es muss dieselbe Codetabelle wie beim Codieren verwendet werden.

- Setze einen Zeiger auf den Wurzelknoten.
- Lese Bit für Bit ein, bis das Ende des Bitstroms erreicht ist.
  - Falls das Bit eine 0 ist, gehe im Codebaum nach links.
  - Falls das Bit eine 1 ist, gehe im Codebaum nach rechts.
  - Erreicht man ein Blatt im Baum, so wird das betreffende Zeichen ausgegeben und der Zeiger wieder auf den Wurzelknoten gesetzt.

Beispiel

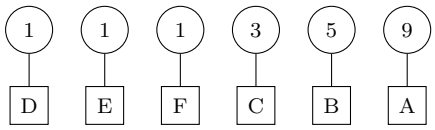
Codiere die Zeichenfolge `AAAAAAAAABBBBBCCCDEF`

Häufigkeitstabelle:

Zeichen :	A	B	C	D	E	F
Häufigkeit:	9	5	3	1	1	1

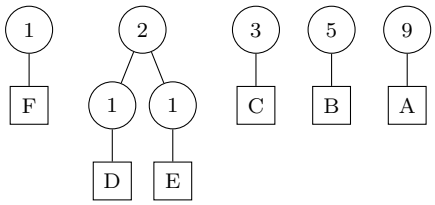
Schritt 1

Stelle die Häufigkeiten der Zeichen als Endknoten („Blätter“) eines sogenannten Baums dar:



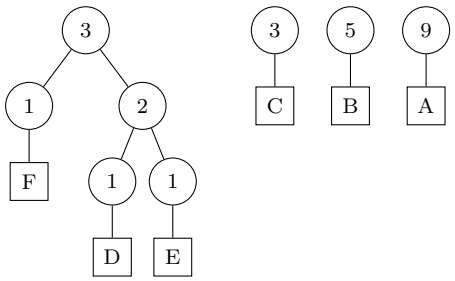
Schritt 2

Die beiden Bäume mit den kleinsten Häufigkeiten (D, E) werden zu einem neuen Baum zusammengefasst und entsprechend ihrer Summenhäufigkeit (1+1=2) einsortiert:



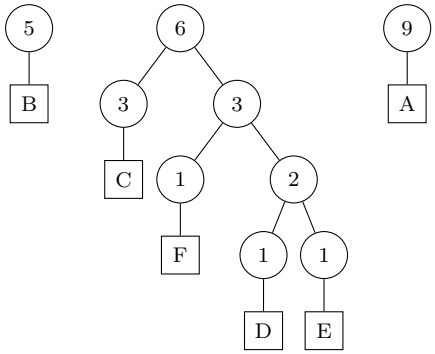
Schritt 3

Die beiden Bäume mit den kleinsten Häufigkeiten werden zu einem neuen Baum zusammengefasst und entsprechend ihrer Summenhäufigkeit (1+2=3) einsortiert:



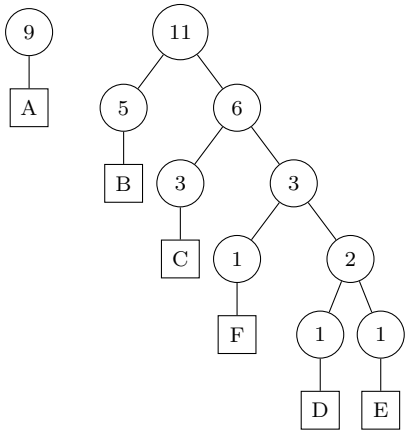
Schritt 4

Die beiden Bäume mit den kleinsten Häufigkeiten werden zu einem neuen Baum zusammengefasst und entsprechend ihrer Summenhäufigkeit (3+3=6) einsortiert:



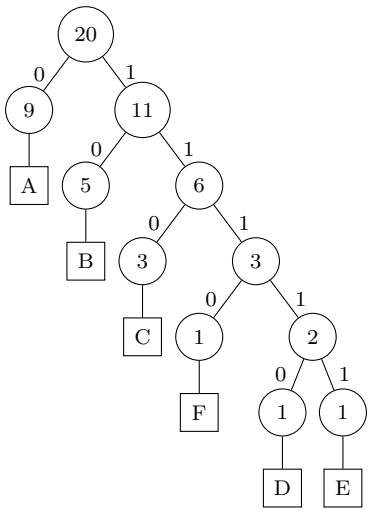
Schritt 5

Die beiden Bäume mit den kleinsten Häufigkeiten werden zu einem neuen Baum zusammengefasst und entsprechend ihrer Summenhäufigkeit (5+6=11) einsortiert:



Schritt 6

Die letzten beiden Bäume werden zusammengefasst und codiert.



`A=0`, `B=10`, `C=110`, `D=11110`, `E=11111`, `F=1110`

Diskussion

Vorteil: Die Huffman-Codierung erzeugt einen Code mit minimaler Redundanz.

Nachteil: Sender und Empfänger müssen dieselbe Häufigkeitstabelle verwenden.

4 LZW

LZW-Eckdaten

- Abraham Lempel, Jacob Ziv und Terry Welch (1984)
- Verbesserung des LZ78-Algorithmus von Lempel und Ziv
- verlustfrei
- patentgeschützt!
- Verwendung:
  - Grafikformat GIF
  - compress (UNIX)

Codierung

Für das Codieren und Decodieren wird eine Basistabelle vorausgesetzt, in der jedes Zeichen des verwendeten Zeichensatzes eine Nummer (Index) hat. Beispielsweise die Tabelle mit den 256 Bytes.

Verwende als Codes zunächst die Zeichennummern aus der Basistabelle. Parallel dazu werden aus den zu komprimierenden Daten Zeichenketten aus zwei (und später mehr) Zeichen gebildet. Diese Zeichenketten werden mit fortlaufenden Nummern in die Basistabelle eingetragen. So können später längere Sequenzen, die zum wiederholten Mal auftreten, durch kürzere Codes (aus der erweiterten Tabelle) dargestellt werden. Während der Codierung werden nur die Codes gespeichert. Die erweiterte Tabelle wird nach Abschluss der Codierung nicht mehr benötigt, da sie der Empfänger in gleicher Weise aufbauen kann.

Decodierung

Beim Decodieren (Dekompression) werden anfangs die Codes gemäss Basistabelle wieder in die entsprechenden Zeichen umgewandelt. Gleichzeitig werden aus den decodierten Zeichen längere Zeichensequenzen gebildet und zur Basistabelle hinzugefügt. Auf diese Weise kann dieselbe Tabelle wie beim Codieren aufgebaut werden, wordurch sich der Prozess selber mit den benötigen Codes versogt.

Beispiel 1

Komprimiere den Text `"GESTRESST"` mit dem LZW-Verfahren.

Basistabelle: `E=0`, `G=1`, `R=2`, `S=3`, `T=4`

Str $\in$ Tab	c: (Str + c) $\notin$ Tab	Code (Binär)	Tab+

Kompressionsgrad:

Beispiel 2

Dekomprimiere den LZW-Code `1, 0, 3, 4, 2, 6, 7`

Basistabelle: `E=0`, `G=1`, `R=2`, `S=3`, `T=4`

Code	Ausgabe	Prognose	Tab+

Originaltext:

Beispiel 3

Komprimiere den Text `"ABABABABC"` mit dem LZW-Verfahren.

Basistabelle: `A=0`, `B=1`, `C=2`

Str ∈ Tab	c: (Str + c) ∉ Tab	Code (Binär)	Tab+

Kompressionsgrad:

Beispiel 4

Dekomprimiere den LZW-Code `0, 1, 3, 5, 1, 2`

Basistabelle: `A=0`, `B=1`, `C=2`

Code	Ausgabe	Prognose	Tab+

Originaltext:

Diskussion des Problemfalls

Im letzten Beispiel gab es zu einem Code-Index noch keinen Eintrag in der erweiterten Tabelle.

Wie decodiert man einen Index, der (noch) nicht in der erweiterten Tabelle steht?

Da der Dekodierungsprozess gegenüber dem Kodierungsprozess immer einen Schritt hinterherhinkt, kann ein „unbekannter“ Index nur dann auftreten, wenn er unmittelbar vorher in die Tabelle aufgenommen wurde.

In welcher Situation kommt dieser Index unmittelbar danach zum Einsatz?

Nur wenn das hinzugefügte Zeichen das erste Zeichen der letzten erkannten Zeichenfolge ist. Schematisch: $ab? \rightarrow aba$

Weil diese Behauptung möglicherweise nicht sofort einleuchtet, zeigen wir, dass das Gegenteil nicht möglich sein kann:

Denn wäre das ergänzte Zeichen zum Beispiel $z \neq a$, so wäre abz in die Tabelle aufgenommen worden. Die nächste Zeichenfolge im Text müsste dann aber mit einem z beginnen,

weshalb der Index von *abz nicht unmittelbar* nach seiner Erzeugung als Code zum Einsatz kommen kann. In diesem Fall wäre der Index auch in der Tabelle des Empfängerprozesses sichtbar, was im Widerspruch zur obigen Annahme steht.

5 JPEG

Übersicht

JPEG ist ein Standard (Joint Photographic Experts Group) für die Komprimierung von Halbtonbildern.

Schritt 1

Transformation der RGB-Eingangsdaten in eine Luminanzkomponente Y und zwei Chrominanzkomponenten Cb (blau/gelb) und Cr (rot/türkis).

RGB Y Cb Cr

Da unser Auge stärker auf Luminanz- als auf Chrominanzunterschiede reagiert, wird eine reduzierte Abtastrate bei den Chrominanzwerten nicht so sehr wahrgenommen wie bei den Luminanzwerten.

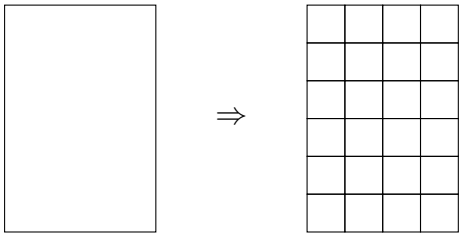
Schritt 2

Durch Mittelwertbildung der Chrominanzwerte in 2×2-Teilblöcken werden die beiden Chrominanzkomponenten auf einen Viertel ihrer Grösse reduziert.

RGB Y Cb Cr

Schritt 3

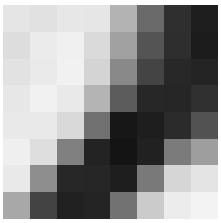
Die Luminanz- und Chrominanzmatrizen werden jeweils in 8×8-Blöcke zerlegt.



Wenn die Anzahl der Bildpunkte in Breite und Höhe kein Vielfaches von 8 sind, müssen zusätzliche Spalten bzw. Zeilen hinzugefügt werden. Der Standard empfiehlt, die letzte Bildspalte bzw. Bildzeile zu wiederholen.

Beispielblock B (aus der Luminanzmatrix):

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	229	225	231	230	179	106	47	30
$y=1$	221	235	240	218	161	84	46	28
$y=2$	226	234	241	212	137	68	40	36
$y=3$	231	241	233	180	92	39	38	48
$y=4$	233	233	217	113	23	30	43	82
$y=5$	239	222	128	35	20	33	123	158
$y=6$	233	142	39	38	30	121	214	228
$y=7$	168	67	33	36	114	203	236	243



Aus verfahrenstechnischen Gründen müssten jetzt von jedem Wert noch 128 subtrahiert werden. Der Übersichtlichkeit wegen, lassen wir diesen Schritt weg.

Schritt 4

Mit der diskreten Cosinustransformation (DCT) wird jeder 8×8-Block B als Summe seiner Frequenzanteile in Form der Matrix F dargestellt.

$$F_{j,i} = \frac{1}{4} \cdot C_i \cdot C_j \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} B_{y,x} \cdot \cos \frac{(2x+1)i\pi}{16} \cdot \cos \frac{(2y+1)j\pi}{16}$$

wobei

C_i, $C_j = 1/\sqrt{2}$ für i, $j = 0$

C_i, $C_j = 1$ sonst

Block B:

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	229	225	231	230	179	106	47	30
$y=1$	221	235	240	218	161	84	46	28
$y=2$	226	234	241	212	137	68	40	36
$y=3$	231	241	233	180	92	39	38	48
$y=4$	233	233	217	113	23	30	43	82
$y=5$	239	222	128	35	20	33	123	158
$y=6$	233	142	39	38	30	121	214	228
$y=7$	168	67	33	36	114	203	236	243

DCT-Transformierte F:

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$j=0$	1110	357	156	−38	2	19	0	0
$j=1$	86	332	−299	−85	18	−13	0	0
$j=2$	58	−216	−17	117	29	9	1	0
$j=3$	−33	63	80	−26	−15	−9	−10	0
$j=4$	3	−3	−53	0	0	0	28	0
$j=5$	12	−7	23	12	15	15	−17	−12
$j=6$	0	0	−9	0	0	−15	0	14
$j=7$	0	0	0	0	0	13	14	−13

Schritt 5

Anschliessend wird die (DCT-)Matrix F elementweise durch eine Quantisierungmatrix Q dividiert. Die Quantisierungsmatrix ist so konstruiert, dass die Elemente mit höheren Frequenzen stärker gedämpft werden als Elemente mit tieferen Frequenzen. Diese Quantisierungstabelle wird vom Standard nicht vorgegeben und steuert den Grad der Kompression und damit auch den Grad des Informationsverlustes.

Eine mögliche Quantisierungstabelle Q:

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	10	12	14	16	19	23	27	32
$y=1$	12	14	16	19	23	27	32	37
$y=2$	14	16	19	23	27	32	37	44
$y=3$	16	19	23	27	32	37	44	52
$y=4$	19	23	27	32	37	44	52	61
$y=5$	23	27	32	37	44	52	61	72
$y=6$	27	32	37	44	52	61	72	85
$y=7$	32	37	44	52	61	72	85	100

DCT-Transformierte F:

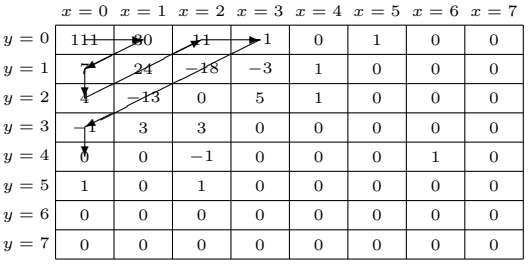
	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$j=0$	1110	357	156	−38	2	19	0	0
$j=1$	86	332	−299	−85	18	−13	0	0
$j=2$	58	−216	−17	117	29	9	1	0
$j=3$	−33	63	80	−26	−15	−9	−10	0
$j=4$	3	−3	−53	0	0	0	28	0
$j=5$	12	−7	23	12	15	15	−17	−12
$j=6$	0	0	−9	0	0	−15	0	14
$j=7$	0	0	0	0	0	13	14	−13

Quantisierte Matrix F_Q:

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	111	30	11	−1	0	1	0	0
$y=1$	7	24	−18	−3	1	0	0	0
$y=2$	4	−13	0	5	1	0	0	0
$y=3$	−1	3	3	0	0	0	0	0
$y=4$	0	0	−1	0	0	0	1	0
$y=5$	1	0	1	0	0	0	0	0
$y=6$	0	0	0	0	0	0	0	0
$y=7$	0	0	0	0	0	0	0	0

Schritt 6

Die quantisierte Matrix wird nach einem Zickzack-Schema linearisiert, das im Bild unten angedeutet ist.



	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	111	30	11	−1	0	1	0	0
$y=1$	7	24	−18	−3	1	0	0	0
$y=2$	4	−13	0	5	1	0	0	0
$y=3$	−1	3	3	0	0	0	0	0
$y=4$	0	0	−1	0	0	0	1	0
$y=5$	1	0	1	0	0	0	0	0
$y=6$	0	0	0	0	0	0	0	0
$y=7$	0	0	0	0	0	0	0	0

111, 30, 7, 4, 24, 11, −1, −18, −13, −1, 0, 3, 0, −3, 0, 1, 1, 5, 3, 0, 1, 0, 0, −1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Diese Liste wird einer Lauflängencodierung unterzogen. (verlustfrei)

Schritt 7

Das Ergebnis der Lauflängencodierung wird zusätzlich mit einer Huffman-Codierung (über alle transformierten Blöcke) komprimiert Auch dieser Schritt ist verlustfrei.

Decodierung (Schritte 1 und 2)

Die Huffman- und die Lauflängencodierung können ohne Informationsverlust wieder rückgängig gemacht werden. Dadurch erhält man die Matrix F_Q zurück.

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	111	30	11	−1	0	1	0	0
$y=1$	7	24	−18	−3	1	0	0	0
$y=2$	4	−13	0	5	1	0	0	0
$y=3$	−1	3	3	0	0	0	0	0
$y=4$	0	0	−1	0	0	0	1	0
$y=5$	1	0	1	0	0	0	0	0
$y=6$	0	0	0	0	0	0	0	0
$y=7$	0	0	0	0	0	0	0	0

Decodierung (Schritt 3)

Multipliziere F_Q elementweise mit der Quantisierungsmatrix Q.

$\Rightarrow$ Matrix F':

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	1110	360	154	−16	0	23	0	0
$y=1$	84	336	−288	−57	23	0	0	0
$y=2$	56	−208	0	115	27	0	0	0
$y=3$	−16	57	69	0	0	0	0	0
$y=4$	0	0	−27	0	0	0	52	0
$y=5$	23	0	32	0	0	0	0	0
$y=6$	0	0	0	0	0	0	0	0
$y=7$	0	0	0	0	0	0	0	0

zum Vergleich die urspüngliche Matrix F:

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$j=0$	1110	357	156	−38	2	19	0	0
$j=1$	86	332	−299	−85	18	−13	0	0
$j=2$	58	−216	−17	117	29	9	1	0
$j=3$	−33	63	80	−26	−15	−9	−10	0
$j=4$	3	−3	−53	0	0	0	28	0
$j=5$	12	−7	23	12	15	15	−17	−12
$j=6$	0	0	−9	0	0	−15	0	14
$j=7$	0	0	0	0	0	13	14	−13

Decodierung (Schritt 4)

Die inverse diskrete Cosinustransformation (IDCT) anwenden. (prinzipiell verlustfrei)

$$B_{y,x} = \frac{1}{4} \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} C_i \cdot C_j \cdot F_{j,i} \cdot \cos \frac{(2x+1)i\pi}{16} \cdot \cos \frac{(2y+1)j\pi}{16}$$

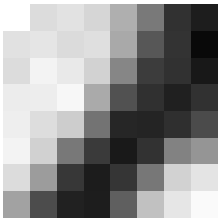
wobei

C_i, $C_j = 1/\sqrt{2}$ für u, $v = 0$

C_i, $C_j = 1$ sonst

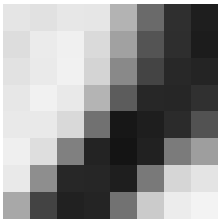
Matrix B':

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	265	220	226	214	175	121	49	29
$y=1$	225	229	219	223	169	87	51	9
$y=2$	220	243	233	210	134	60	49	24
$y=3$	236	237	247	170	81	48	32	49
$y=4$	238	221	205	115	38	36	47	75
$y=5$	243	209	120	58	25	50	130	149
$y=6$	220	156	55	28	53	119	210	228
$y=7$	162	76	33	29	95	193	230	250



Zum Vergleich: B

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$y=0$	229	225	231	230	179	106	47	30
$y=1$	221	235	240	218	161	84	46	28
$y=2$	226	234	241	212	137	68	40	36
$y=3$	231	241	233	180	92	39	38	48
$y=4$	233	233	217	113	23	30	43	82
$y=5$	239	222	128	35	20	33	123	158
$y=6$	233	142	39	38	30	121	214	228
$y=7$	168	67	33	36	114	203	236	243



Decodierung (Schritte 5–7)

- Die einzelnen Luminanz- und Chrominanzblöcke werden wieder zu einem Luminanzbild und zwei Chrominanzbildern zusammengesetzt.

  Die verfahrensbedingten hinzugefügten Zeilen und Spalten müssen jetzt natürlich wieder entfernt werden.

- Die (kleineren) Chrominanzbilder werden durch Vervierfachen jedes Farbwerts wieder auf dieselbe Grösse wie das Luminanzbild gebracht.

- Die Luminanz- und Chrominanzbilder werden in ein RGB-Bild zurücktransformiert.