JPEG2000

Seminar Datenkompression

Universität Bonn, WS 2005/2006 Prof. Dr. N. Blum, Matthias Kretschmer

Gereon Schüller
Email: `X@Y mit X='schuelle', Y='cs.uni-bonn.de'`

29. November 2005

1 Probleme von JPEG

- Bildung von Block-Artefakten
  Erläuterung an Hand von Grafik auf Folie, ggf. zeigen der Blöcke falls auf Beamer schlecht erkennbar. Die Blöcke rühren vom „Tiling“ für die DCT her, die die Grafik in 8×8-Blöcke zerschneidet, um Laufzeit klein zu halten, die für DCT stets $\Theta n \log n$ beträgt.[1]

- Schlechtes Ergebnis bei Texten und Computergrafiken
  Zeigen der „Wolkenbildung“ zwischen den Buchstaben bei JPEG-komprimiertem Text, Buchstaben erzeugen raschen Kontrastwechsel (hohe Frequenz), DCT muss immer kleinere Kosinuswellen zur Approximation verwenden → Grey-Effekt

- Nicht robust gegen Dateifehler
  Ein Byte mittels Hex-Editor verändert führt zu Farbfehler in einem Block, restliches Bild ist gegenüber dem ersten Abschnitt um einige Pixel verschoben

- Dateigröße auf 64.000×64.000 Pixel beschränkt
  Nachteil ist eher theoretischer Natur, aber immerhin ist Größenbeschränkung immer Hindernis in der Praxis.

2 Anforderungen an JPEG2000

- Komprimierung für alle Typen von Grafiken (Fotos, computergenerierte Grafiken, Texte etc.)

- Anpassbar für viele Übertragungswege (Echtzeit, limitierte Bandbreite, Speicherung in Archiven)
  JPEG wurde zu einer Zeit entwickelt, als nur Unternehmensnetzwerke existierten, Internet kaum Verbreitung hatte und Bilder nicht in Echtzeit zur Verfügung stehen mussten. Heute Anzeige von Grafiken per Internet, WLAN, auf UMTS-Handy, gerade bei Funkweg wäre es einfacher, Übertragungsfehler nicht zwangsläufig korrigieren zu müssen

- Bessere Qualität bei gleicher Dateigröße als bisherige Standards

- Schaffung von verlustlosem Kompressionsalgorithmus
  Im Vortrag wird nur die verlustbehaftete Komprimierung behandelt, da die verlustlose meist nur Schritte weglässt. Da wo es wichtig ist, wird auf die Unterschiede hingewiesen.

Idee:

[1] In dieser Ausarbeitung ist Text auf Folien schwarz, Redetext blau, Tafelanschrift grün und Programmvorführung braun gekennzeichnet.

- Verwendung von diskreter Wavelet-Transformation statt diskreter Kosinus-Transformation
  DWT hat gegebenüber DCT den Vorteil, dass Laufzeit geringer und hoher Frequenzwechsel nur lokal auf Anzahl der Koeffizientzen wirkt, daher bessere Wirkung bei Texten

- Komprimierung mittels Entropieenkodierung

- Möglichkeit reversibler Komprimierung
  JPEG ließ zwar reversible Komprimierung zu, diese wurde aber nur selten verwendet und von vielen Encodern nicht unterstüzt. Wichtig z.B. für Röntgenbilder

- Speicherung in Blöcken, so dass Beschädigungen nur lokal wirken
  Natürlich können größere Verfälschungen nicht verlustlos bleiben, aber dann wegen Header-Nummerierung nur lokale Zerstörung der Grafik

3 Ablauf der Enkodierung

Erläuterung des Verlaufsdiagrammes auf der Folie, *kurze* Erläuterung der Punkte

- Dekodierung im Wesentlichen umgekehrt
  Dekodierung wird im Vortrag nicht gesondert behandelt, da Schritte im Wesentlichen Umgekehrt

4 Pre-Processing

1. „Tiling“: Zerschneiden des Bildes in kleinere Stücke, falls Arbeitsspeicher nicht ausreicht
   Auf speicherschwachen Geräten kann das Bild in Tiles zerlegt werden, falls der Arbeitsspeicher zu gering ist. Im Folgenden wird davon ausgegangen, das die Grafik nicht zerlegt wird

2. „Level Offset“: Addieren von -2^{B-1}, falls Bildwerte als positive Integerwerte vorliegen
   Für Wavelet-Transformation ist es günstig, wenn die Werte um die 0 herum zentriert sind. Manche Quellformate kennen aber keinen negativen Werte, sondern nur positive. Also wird der Wertebereich einfach verschoben. Dieser Schritt ist natürlich verlustlos.

3. „Irreversible Color Transform“: Umwandeln des Bildes aus dem RGB-Farbraum in den YC_bC_r-Farbraum

Farbtransformation nach YC_bC_r

- Ursprünglich Entwicklung aus der Fernsehtechnik zur Darstellung von Farbbildern
- Bild wird in seine Luminanz, d.h. Helligkeit (=s/w-Bild) und seine Farbkomponenten (=Differenz zu blau und rot) zerlegt
  Als das Farbfernsehen eingeführt werden sollte, war es wichtig, die Farbinformation getrennt von den Helligkeitswerten zu übertragen. Aus diesem Grund wird ein psychologisch-physisches Modell zugrunde gelegt, dass die Luminanz so errechnet, wie es der Mensch wahrnehmen würde, daher stammen die Matrixeinträge. Man sieht ferner, dass sich die Werte der Ersten (Y-)Zeile zu 1 addieren.

Mathematisch:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.114 \\ -0.163 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Falls erforderlich wird an der Tafel dargelegt, warum die Matrix invertierbar ist. Prinzipiell sollte dies aber für die Hörer aus dem Grundstudium bekannt sein

- Die Matrix ist invertierbar, aber die Transformation wegen Rundung auf Integerwerte irreversibel
- Für verlustlose Komprimierung alternative Matrix mit ganzzahligen Einträgen

Vorteil für unsere Zwecke: Dynamischer Bereich der Farbkompenenten „flacher“, daher besser komprimierbar
Das Verfahren wird mittels eines selbstgeschriebenen Programms visualiert. Erst wird eine Grafik in RGB zerlegt. Man erkennt dann, dass die Bilder einen hohen dynamischen Bereich haben, aber jeder Farbauszug in einigen Bereichen seine Schwächen, so dass er nicht als s/w-Grafik geeignet wäre. Danach Zerlegung gem. YC_bC_r. Der Y-Auszug wirkt natürlich. Wenn man nur ein s/w-Bild dekomprimieren möchte, reicht es, diesen zu übertragen. Ferner erkennt man die geringe Dynamik der Differenzauszüge. Im Weiteren wird nur noch der Y-Auszug betrachtet, da alle Bilder gleichartig komprimiert werden.

5 Diskrete Wavelettransformation

- Herzstück von JPEG2000
- 2 Wavelets stehen zur Auswahl:
  1. Daubechies-5/3-Wavelet für verlustfreie Kompression (auch „LeGall“ genannt)

2. Daubechies-9/7-Wavelet für verlustbehaftete Kompression
   Erinnerung an den vorherigen Vortrag, die Eigenschaften der Daubechies-Wavelets

Vorgehen:

- Die Zeilen und Spalten des Bildes werden mittels eines Hoch- und eines Tiefpassfilters gefiltert
- Anschließen werden Zeilen und Spalten einem „Downsampling“ unterzogen, d.h. jeder zweite Wert wird gelöscht
  Vorführung eines Hoch- und Tiefpassfilters mit Downsampling anhand des Programms, Erläuterung des zu sehenden Effekts

⇒ Dabei ist das Filtern nach Zeilen und Spalten kommutativ
Es ist also gleichgültig, ob zuerst nach Spalten oder zuerst nach Zeilen gefiltert wird. Unterstützend dazu Blockgrafik auf der Folie

- Das Verfahren wird rekursiv bis zu 32 mal angewendet (für natürliche Bilder 4 bis 8 mal)
  Die Rekursion wird durch das Bild auf der Folie visualisiert, welches fortlaufend in die vier Quadranten LL/LH/HL/HH geteilt wird
  Hinweis darauf, das man bei dem HH-gefilterten Quadraten die Details noch erkennen kann, aber die Anzahl der Samples quadratisch abnimmt. Erläuterung der Grafik auf der Folie. Erinnerung an die Orthogonalität der Daubechies-Wavelets. Falls im vorigen Vortrag nicht geschehen, Hinweis darauf, dass die DWT eine Vektorfaltung ist die in Zeit $\Theta(n)$ möglich ist.

6 Quantisierung

Die berechneten Wavelet-Koeffizienten liegen nun als Dezimalzahlen vor und müssen als Integerwerte gespeichert werden

- Es wird ein sog. *Quantifizierer* eingesetzt. Dabei werden die Werte folgt gerundet:
$$q = sign(y) \left\lfloor \frac{|y|}{\Delta_b} \right\rfloor$$

⇒ Das Intervall $[-\Delta_b; \Delta_b]$ wird auf 0 abgebildet, daher werden besonders kleine Werte sehr effektiv komprimiert

- Dieses Intervall wird auch „Todeszone“ *(deadzone)* genannt.
  Tafelbild: Zahlengerade mit Intervallen Δ_b, in der Mitte 0, Visualisierung der Abbildung über Pfeile, Erläuterung, dass immer „zur 0 hin“ gerundet wird, das Vorzeichen durch die Multiplikation mit $sign(y)$ erhalten bleibt
  Erläuterung, das sehr viele Werte durch den Highpass-Filter ≈ 0 sind

und daher der Deadzone-Quantisierer diese Werte ausblendet (Rauschunterdrückung). Ggf. nochmaliges Zeigen der Programm-Ausgabe. Bei der verlustlosen Komprimierung wird freilich auf die Quantisierung verzichtet. Durch die perfekte Wavelet-Zerlegung ist dann aus Rückrechnung der rationalen Wavelet-Koeffizienten die verlustlose Kompression möglich.

7 Eingebettete Block-Kodierung

Die Block-Kodierung dient der eigentlichen Komprimierung der Wavelet-Koeffizienten. In den vorigen Folien wurde gezeigt, dass durch die DWT die Datenmenge noch nicht verringert hat, sondern nur darauf vorbereitet hat, die Daten besser komprimieren zu können

- Die quantisierten Koeffizenten werden nun in „Blöcke“ à 32×32 Werte eingeteilt

$\Rightarrow$ Idee: Die Blöcke werden einzeln kodiert und gespeichert, dann sind bei Beschädigungen nur kleine Bereiche betroffen
Erinnerung an die Eingangsidee, Übertragungsfehler aufzufangen.

- Jeder Block wird Streifenweise nach Spalten von links nach rechts gescannt, wobei die Spalte nach 4 Werten in die nächst Spalte gewechselt wird.
Tafelbild: Scanprinzip mittels Pfeilen, Bits symbolisiert als Kreise. Die Pfeile fangen oben links an und gehen sägezahnartig durch die Blöcke.

- Dabei werden die einzelnen Werte nicht auf Byte- sondern auf Bitebene betrachtet. Jede Bit-Stelle wird als eigene Ebenene *(bit-plane)* betrachtet. Dabei wird zuerst die Ebene mit der größten Stelle (MSB-Plane) kodiert
Anschreiben einer 8-Bit-Zahl, z.b. 0010011, Erläuterung was mit MSB-Plane gemeint ist.

8 Kontext-Signifikanz-Kodierung

Die *Kontext-Signifikanz* eines Bits berechnet sich aus den Signifikanzen σ seiner horizontalen, vertikalen und diagonalen Nachbarn. Dabei ist:

$$K^h[j] = \sigma[j_r, j_c - 1] + \sigma[j_r, j_c + 1] \tag{1}$$

$$K^v[j] = \sigma[j_r - 1, j_c] + \sigma[j_r + 1, j_c] \tag{2}$$

$$K^d[j] = \sum_{k_r=\pm 1} \sum_{k_c=\pm 1} \sigma[j_r + k_r, j_c + k_c] \tag{3}$$

Tafelbild: 9 Bits quadratisch angeordnet. Mittels Linienzug Umhüllung von K^h, K^v, K^d. Falls möglich mit verschiedenfarbiger Kreide, sonst Unterscheidung mittels Schraffur o.ä.
K^{sig} errechnet sich aus K^h,K^v,K^d.

9 1. Durchlauf: Signifikanz-Propagation

Kodiere alle insignifikanten Bits, die kontext-signifikant sind. Dabei seien
$v[j]$:=Wert von Bit an Stelle j
$\sigma[j]$:=Signifikanz von Bit an Stelle j, inital 0
$\pi[j]$:=Markierung, dass Bit schon kodiert wurde;
falls der Wert des Bits 1 ist, so kodiere sein Vorzeichen und setze die Signifikanz. Dieser Durchlauf heißt Signifikanz-Propagierung, weil hier erstmals festgelegt wird, welche Bits überhaupt kontext-signifikant sein sollen. Gleichzeitig werden aber auch schon Bits kodiert, wobei MQ-ENCODE eine arithmetische Kodierung ist, ähnlich der die schon in einem der vorhergehenden Vorträge behandelt wurde und daher hier nicht mehr erläutert wird. Die Vorzeichen werden dabei gesondert kodiert, da diese für die Dekodierung natürlich besonders wichtig sind.

```
1:  for all j do
2:      if σ[j] = 0 ∧ K^sig > 0 then
3:          MQ-ENCODE(v[j], K^sig[j])          ▷ Kodiere Sample arithmetisch
4:          if v[j] = 1 then
5:              σ[j] ← 1                        ▷ Setze Signifikanz
6:              ENCODESIGN(x)                   ▷ Kodiere Vorzeichen
7:          end if
8:          π[j] ← 1                            ▷ Markiere Bit als kodiert
9:      end if
10: end for
```

Algorithm 1: Signifikanz-Propagation

Erläuterung des Algorithmus: Der Algorithmus läuft über alle Bits und überprüft im ersten IF-Statement Signifikanz und Kontext-Signifikanz. Nur wenn das Bit auch gesetzt ist, wird seine Signifikanz gesetzt und das Vorzeichen kodiert.

10 2. Durchlauf: Magnituden-Verfeinerung

Dieser Durchlauf dient dazu, die im vorherigen Durchlauf kodierte Bit-Ebene zu verfeinern. Jedes Bit ist jetzt halb so viel wert wie das vorherige. Die Kontext-Signifikanz hängt jetzt nicht mehr nur von den benachbarten Bits, sondern vor allem von der vorhergehenden Bit-Stelle ab. Daher wurde der Magnituden-Kontext eingeführt.
Kodiere nun alle Bits, die erst im vorherigen Signifikanz-Propagierungs-Durchlauf signifikant geworden sind, und kodiere zusätzlich den *Magnituden-Kontext* wobei

$$K^{mag}[j] = \begin{cases} 0 & \text{falls } v^{(p+1)}[j] = 1 \wedge K^h[j] + K^v[j] + K^d[j] = 0 \\ 1 & \text{falls } v^{(p+1)}[j] = 1 \wedge K^h[j] + K^v[j] + K^d[j] > 0 \\ 2 & \text{falls } v^{(p+1)}[j] > 1 \end{cases}$$

```
1: for all j do
2:     if σ[j] = 1 ∧ π[j] = 0 then
3:         K^mag = ...
4:         MQ-ENCODE(v[j], K^mag[j])
5:     end if
6: end for
```

11 Phase 3: Aufräumphase

Bisher wurden noch nicht alle Bits kodiert, alle verbliebenene Bits werden nun kodiert. Da nun viele Bits insignifikant sind und gar nicht mehr einzeln komprimiert werden müssen, existiert ein sog. Run-Mode, in dem einfach die Anzahl der +bersprungenen Bits vermerkt wird. Bei der Dekrompression bleibt die Reihenfolge dieser Phasen gleich und die Bildwerte werden folglich immer feiner

- Kodiere nun alle verbliebenen Bits (die nunmehr insignifikant sein müssen)
- Falls in einem Streifen 4 oder mehr Bits sowie ihre Nachbarn insignifikant sind, schalte in den sog. *Run-Mode* und vermerke lediglich diese Tatsache

12 Dekodierung

- Verläuft prinzipiell genau umgekehrt
- Neu bei JPEG2000: Einzelne Schritte können übersprungen werden, z.B.
  - Farbtransformation für s/w-Ausgabe
    Es ist nicht nötig, die Farbe z.B. für Ausgabe im Zeitungsdruck wieder hinzuzufügen, um sie dann wieder auszublenden.
  - Dekompression von Layern mit niedriger Qualität, z.B. für kleine Displays
    Es ist auch nicht nötig, die beste Qualtät zu erreichen, wenn das Ausgabemedium sie gar nicht mehr darstellen kann, z.B. auf Handys, bei denen die Übertragungsrate knapp ist. Mehr Qualität kann nachgefordert werden, außerdem ist es möglich, erst „Thumbnails“ anzusehen. Es entfällt die bisher nötige manuelle Runterrechnung auf kleine Bildgrößen. Die Bit-Streams werden dafür speziell organisiert, was aber mit theoret. Informatik wenig zu tun hat.
- Weiterhin können Bildbereiche weniger stark komprimiert werden, sog. *Regions Of Interest (ROI)*. Dazu wird zu den Koeffizienten ein Offset addiert
  Gesichter z.B. werden mit einem Offset versehen. Dadurch gehen die Details weder in Quantisierung noch in der Block-Kodierung, die Bereiche bleiben Detailreicher.

- Da jeder Block einzeln kodiert wird, gehen bei Übertragungsfehlern höchstens kleine Bildinformationen verloren

13 Beispiele

Text vom Anfang als JPEG2000 komprimiert

Man erkennt an dem Beispieltext, dass keine Wolken mehr vorhanden sind, obwohl beide Grafiken die gleiche komprimierte Größe haben
Vergleich JPEG zu JPEG2000

Man erkennt eindeutig in der Mustergrafik, dass die Farbabstufung im fotografierten Himmel erhalten blieb, und keine Blocks gebildet wurden. Das gesamte Bild wirkt schärfer. [Falls noch Zeit, kann man das Bild am Notebook vergrößern.]

14 Quellen

1. Gray, Karen L.: „The JPEG2000 Standard“, TU München, o.J.
2. Marcellin, Michael W.: „JPEG2000“, University of Arizona, Tucson, Dezember 2002
3. Rabbani, Majid: „JPEG-2000: Background, Scope, And Technical Description“, Eastman Kodak Research Laboratories, Rochester NY, Dezember 1998
4. Taubman, D. et. al.: „Embedded Block Coding in JPEG2000“, HP Laboratories, Palo Alto, Februar 2001
   Taubmann hat die Block-Kodierung erfunden...