

Videokompression am Beispiel H.264

Medientechnologie IL

Andreas Unterweger

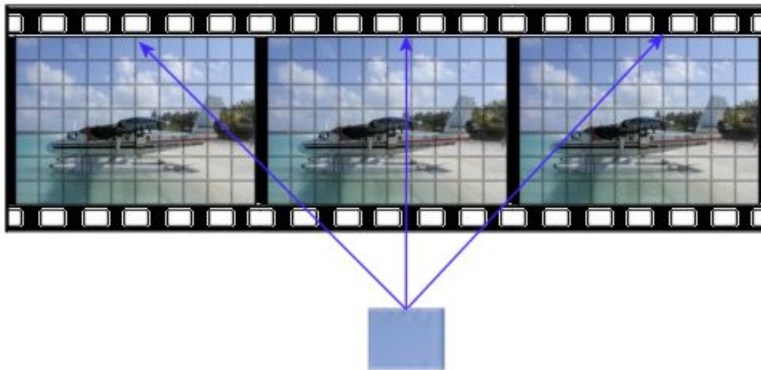
Vertiefung Medieninformatik und Bildverarbeitung
Studiengang ITS
FH Salzburg

Sommersemester 2022

- Gehirn interpretiert Bewegungen durch Helligkeitsänderungen
 - Bewegung eines Objektes bewirkt Bewegung der dazugehörigen (vom Objekt reflektierten) Photonen auf der Netzhaut (den Rezeptoren)
 - Rezeptoren können ca. 10 Änderungen pro Sekunde wahrnehmen
 - Mehr als ca. 16 Änderungen pro Sekunde bewirken Eindruck von Bewegung (schwer von tatsächlicher Bewegung unterscheidbar)
- Aufeinander folgende Bilder können Bewegungsbildeindruck hervorrufen
- Zusätzlich: Nachbildwirkung (**nicht** Ursache für Bewegungsbildeindruck)
 - Rezeptoren „leuchten“ ca. 40 ms nach (Reizamplitude klingt ab)
 - Neue und alte Reize überlagern einander → Glättung (Tiefpassfilter)
 - Eindruck (teilweise) „flüssiger“ Bewegung
- Verschiedene Voraussetzungen für tatsächlich „flüssige“ Bewegtbilder:
<https://www.red.com/red-101/high-frame-rate-video>

Zeitliche Korrelation in Bewegtbildern

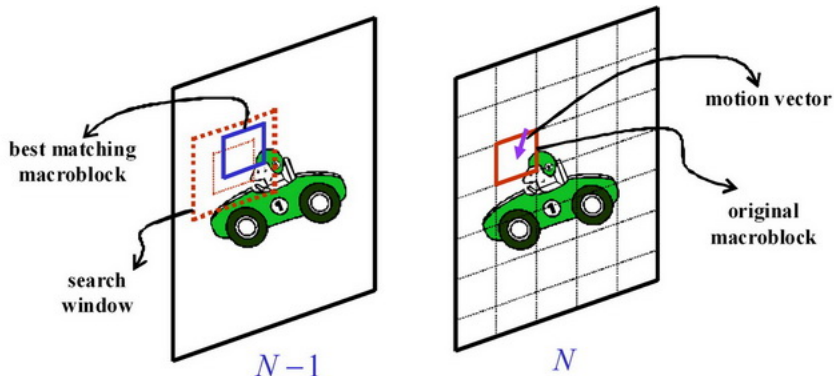
- Aufeinander folgende Bilder korrelieren meist stark
- Ausnutzen durch spezielle Differenzkodierung



Quelle: http://www.labdv.com/images/library/dv_basics/mpeg_explained/tempcompress.jpg

- Bewegungen zwischen Bildern betreffen nicht alle Pixel gleichermaßen
- Pixelweise Bewegungssuche ist zu aufwändig
- Blockbasierte Bewegungssuche (englisch *Motion Estimation*, ME)
 - Vorgehensweise: Passenden Block suchen (englisch *Block Matching*)
 - Bewegungsvektor (englisch *Motion Vector*): Versatz zwischen Ausgangs- und gefundenem Block (X- und Y-Versatz)
 - Parameter: Suchradius (Fenstergröße)
 - Vergleichsmetrik notwendig (implementierungsabhängig)
- Speicherung: Statt komplettem Block
 - Ein Bewegungsvektor (mit zwei Komponenten)
 - Pixelweise Differenz zwischen Ausgangs- und gefundenem Block (englisch *Block Motion Compensation*, kurz BMC oder MC)

Motion Estimation II



Quelle: <http://www-sipl.technion.ac.il/UploadedFiles/BlockBasedMotionEstimation.jpg>

- Summe von Pixeldifferenzen ist keine brauchbare Metrik
 - Vorzeichenverschiedene Differenzen heben sich auf
 - Weißes Rauschen ist quasi immer so gut wie ein perfekter Treffer
- Andere Blockvergleichsmetriken notwendig
- Notation:
 - $M, N \in \mathbb{N}^+$: Blockdimensionen in Pixel
 - $m \in \{x \in \mathbb{N} \mid 0 \leq x \leq M - 1\}$, $n \in \{x \in \mathbb{N} \mid 0 \leq x \leq N - 1\}$
 - $X_{m,n}$: Pixel des Ausgangsblocks
 - $X'_{m,n}$: Pixel des Vergleichsblocks
 - x_{max} : Maximal möglicher Pixelwert (z.B. 255 bei 8 Bit)
 - d_{Metrik} : Differenzwert (Metrikausgabewert)
- Praktische Einschränkung: Nur Pixel des Y-Kanals werden betrachtet
 - $X_{m,n}$ und $X'_{m,n}$ sind Zahlen und keine Vektoren

- Sum of Absolute Differences (SAD):

$$d_{SAD} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |X'_{m,n} - X_{m,n}|$$

- Sum of Squared Differences (SSD):

$$d_{SSD} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (X'_{m,n} - X_{m,n})^2$$

- Mean Squared Error (MSE):

$$d_{MSE} = \frac{d_{SSD}}{MN} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (X'_{m,n} - X_{m,n})^2$$

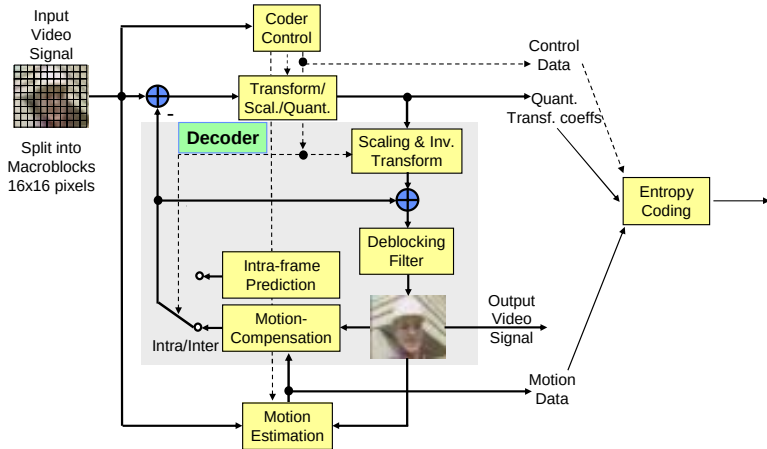
- Peak Signal-to-Noise Ratio (PSNR):

$$d_{PSNR} = 10 \log \left(\frac{x_{max}^2}{d_{MSE}} \right) = 20 \log \left(\frac{x_{max}}{\sqrt{d_{MSE}}} \right), \quad [d_{PSNR}] = \text{dB}$$

- Weiterer Parameter: Anzahl zu durchsuchender Vorgängerbilder
 - Vorteil: Höhere Effizienz durch möglicherweise bessere Treffer
 - Nachteil: Speicherung der Referenzbildnummer notwendig
 - Weiterer Nachteil: Lineare Erhöhung der Suchdauer
 - Volle Suche ist aufwändig (quadratische Komplexität!)
- Nicht alle Möglichkeiten ausprobieren
- Nachteil: Findet eventuell ein lokales Minimum (nicht das globale)
 - Dutzende ME-Algorithmen in der Literatur

Überblick: Typischer H.264-Encoder

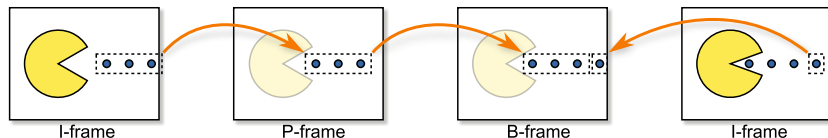
- Typische Encoder-Architektur (Standard gibt nur Dekodierung vor)



Quelle: Wiegand, T. and Sullivan, G. J.: The H.264 | MPEG-4 AVC Video Coding Standard.
http://ip.hhi.de/imagecom_G1/assets/pdfs/H264_03.pdf. 2004.

Übersicht Bildtypen

- Vereinfacht: Drei Bildtypen (nach Kodierungsabhängigkeit)
 - **Intra**: Unabhängig von anderen Bildern
 - **Prädiziert**: Verwendet nur vorangegangene¹ Bilder (MC)
 - **Bidirektional**: Verwendet vorangegangene und folgende Bilder (MC)
- Kodierungsentscheidung pro (Makro-)Block ($16 \cdot 16$ Y-Pixel):
 - I-Block: In I-, P- und B-Bildern erlaubt
 - P-Block: In P- und B-Bildern erlaubt
 - B-Block: In B-Bildern erlaubt
- Encoder darf Blocktypen (mit diesen Einschränkungen) frei wählen

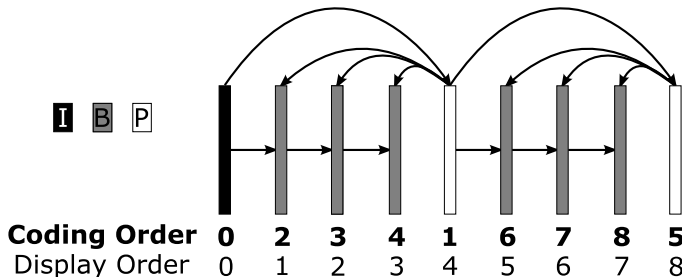


Quelle: <http://en.academic.ru/dic.nsf/enwiki/386020f>

¹Theoretisch möglich (unüblich): auch folgende Bilder

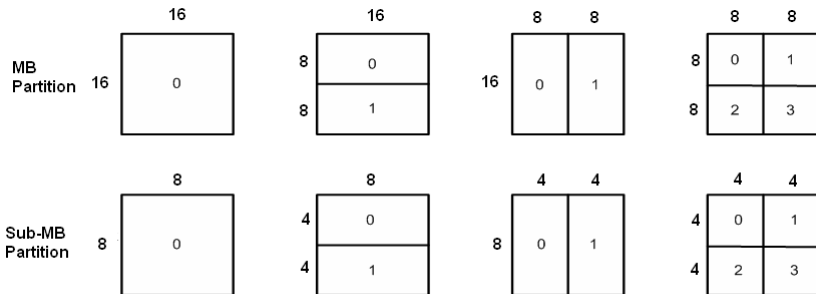
Besonderheiten bei der Verwendung von B-Frames

- MC aus folgenden („zukünftigen“) Bildern ist unpraktikabel
- Zum Kodieren Bilder umsortieren → Abweichende Kodierreihenfolge (englisch *Coding Order*); Beispiel: IBBbPBBbP
- Für MC verwendete Bilder müssen zuerst kodiert werden
- Wichtig: Darstellungsreihenfolge (englisch *Display Order*) muss beibehalten werden



Interprädikation (englisch *Inter Prediction*) I

- Makroblöcke ($16 \cdot 16$ Y-Pixel) können partitioniert werden
- $8 \cdot 8$ Makroblockpartitionen können subpartitioniert werden
- Jede Partition kann auf unterschiedliche Referenzbilder verweisen
- Jede (Sub-)Partition kann unterschiedliche Bewegungsvektoren haben



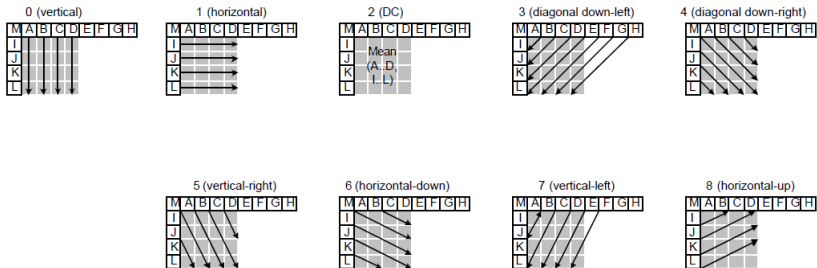
Quelle: Kwon, S., Tamhankar, A. and Rao, K.R.: Overview of H.264 / MPEG-4 Part10.

http://www.powershow.com/view/19bc9-NDQyM/Overview_of_H_264_MPEG4_Part10_powerpoint_ppt_presentation. 2004.

- Subpixelgenauigkeit bei Bewegungsvektoren möglich
- Interpolation notwendig
 - Realisierung: Interpolationsfilter (ohne Details)
 - 1. Schritt: Pixel an „halben Positionen“ aus Pixeln interpolieren
 - 2. Schritt: Pixel an „Viertel-Positionen“ aus „halben“ interpolieren
- Weiterer ME-Parameter → Komplexitätserhöhung
- Praktische Maßnahmen:
 - Schnelle ME-Algorithmen
 - Kleinerer ME-Radius
 - Geringe Anzahl Referenzbilder
- B-Blöcke mitteln **zwei unabhängige** Treffer vor MC
- Weitere Komplexitätserhöhung

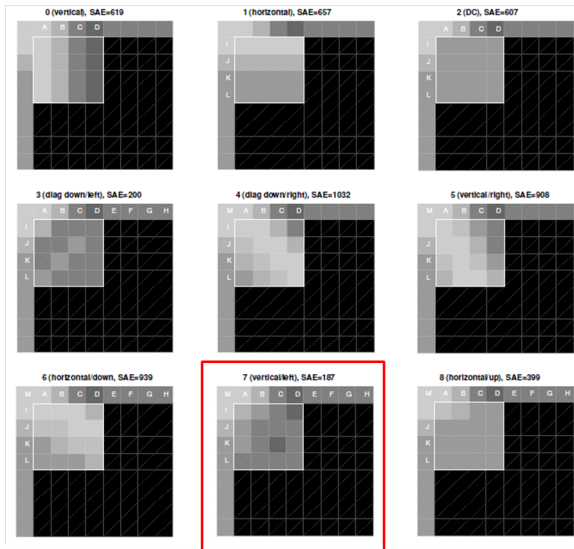
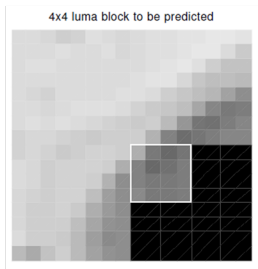
Intraprädikation (englisch *Intra Prediction*) I

- I-Block-Kodierung ähnlich wie in JPEG (mit Verbesserungen)
- Wahl: 1 $16 \cdot 16$ -Block oder $16 \cdot 4 \cdot 4$ -Blöcke (später $8 \cdot 8$ ergänzt)
- Blöcke werden aus Nachbarblöcken extrapoliert $\rightarrow$ Differenzkodierung
- Extrapolationsrichtung pro Block wählbar (neun Modi bei $4 \cdot 4$)



Quelle: Richardson, I. E. G.: H.264 / MPEG-4 Part 10: Intra Prediction. <http://www.vcodex.com/h264.html>. 2003.

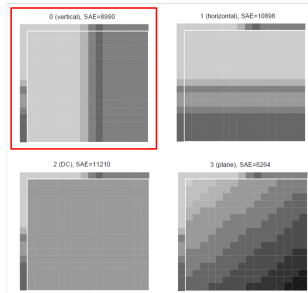
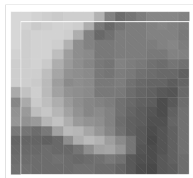
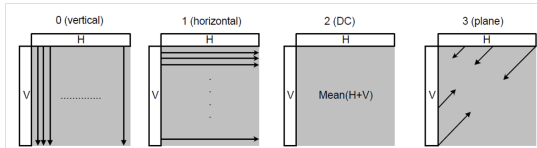
Intraprädikation (englisch *Intra Prediction*) II



Quelle: Richardson, I. E. G.: H.264 / MPEG-4 Part 10: Intra Prediction. <http://www.vcodex.com/h264.html>. 2003.

Intraprädikation (englisch *Intra Prediction*) III

- Bei $16 \cdot 16$ -Blöcken: Vier Richtungen (drei ident zu $4 \cdot 4$)

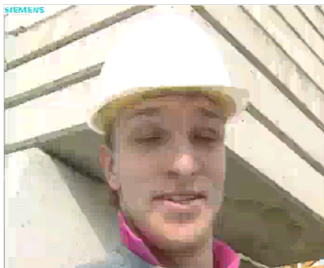


Quelle: Richardson, I. E. G.: H.264 / MPEG-4 Part 10: Intra Prediction. <http://www.vcodex.com/h264.html>. 2003.

- Quantisierungsmatrix (JPEG-ähnlich) mit Skalierung (AAC-ähnlich)
 - Höhere Frequenzen werden durch Skalierung stärker quantisiert
 - Koeffizientenskalierung automatisch (später einstellbar)
 - Quantisiererschrittweite unabhängig von Skalierung (Parameter Q_{step})
- Quantisierungsparameter QP
 - Bestimmt Quantisiererschrittweite Q_{step} (Divisor)
 - Wertebereich 1 bis 51 (später: 0 bis 51)
 - $QP = 6 \rightarrow Q_{step} = 1,25$
 - $QP + 6 \rightarrow 2Q_{step}$ bzw. $QP + 1 \rightarrow \sqrt[6]{2}Q_{step}$
 - $QP - 6 \rightarrow \frac{Q_{step}}{2}$ bzw. $QP - 1 \rightarrow \frac{Q_{step}}{\sqrt[6]{2}}$
 - Pro Makroblock einstellbar

Deblocking-Filter

- Adaptives Tiefpassfilter zur Glättung von Blockkanten
 - Berücksichtigt Makroblockmodi und -kantenpixel
 - Erhält hinreichend harte Kanten (Schärfe)
 - Stärke zusätzlich pro Videosequenz einstellbar (auch deaktivierbar)
- Zumeist subjektive Qualitätsverbesserung
 - Wird nach dem (De-)Kodieren jedes Bildes angewandt
- P- und B-Prädiktion auf gefilterten Bildern



Danke für die Aufmerksamkeit!

Fragen?