Motivation

Worum geht es?

- Bild- und Videodaten abspeichern
- Schneller Zugriff
- Schnelle Übertragung
- Geringer Platzbedarf
- Hohe Bildqualität

Dazu brauchen wir:

- Einheitliches Datenformat
- Bild- und Videokomprimierung

Womit haben wir es zu tun?

- Bild- und Videodaten sind gegeben als zwei- bzw. dreidimensionales Array aus **Pixeln** (Bildpunkten). Die dritte Dimension ergibt sich bei Videodaten aus der Zeit $\Rightarrow$ Abfolge von Bildern. Diese Einzelbilder heißen **Frames**.
- Ein Pixel besteht entweder aus einem Grauwert (bei schwarz-weiß) oder drei Farb**komponenten**.
- Eine Abfolge von Frames heißt Video-**Sequenz**.
- Frames können im **Interlaced**-Modus übertragen werden. Dabei werden für jeden Frame zuerst die geraden Zeilen $(0, 2, 4, \ldots)$ übertragen und dann die ungeraden Zeilen. (**odd** und **even fields**)
- In neumodischen Bild- und Videoformaten gibt es noch weitere Unterteilungen: Objekte und Hintergrund werden als eigene **video object planes** (**VOP**) unabhängig voneinander kodiert.

Farbenlehre

- Was ist Farbe? Elekromagnetische Wellen mit Wellenlängen im Bereich 400nm bis 700nm.
- Wieso genügen drei Komponenten um eine Farbe zu bestimmen? Weil das Auge auf nur drei Stimuli reagiert.
- **Gamma Correction**: Nichtlineares Helligkeitsempfinden des Menschen (doppelt so starke Lichtquelle weniger als doppelt so hell) wird beim Digitalisieren nachvollzogen durch eine Potenzfunktion (Potenz meistens 0.45). Das Ergebnis sollte dann mit z.B. R' statt R bezeichnet werden. Wird aber nicht $\Rightarrow$ Durcheinander.
- **Uniformität** eines Farbraums bedeutet, dass bei Veränderung einer Komponente um einen gewissen Betrag sich ein bestimmter subjektiver Effekt einstellt der unabhängig von der aktuellen Farbe ist.
- Eine gewisse Intensität einer Lichtquelle empfindet der Mensch als blaue Farbe am dunkelsten und als grüne Farbe am hellsten. Deshalb sollte die Helligkeit (Luminanz) immer gewichtet berechnet werden.
- Das Auge reagiert bezüglich Bilddetails und Bewegung auf die Intensität weit stärker als auf Farbe. Daher sollte die Farbinformation stärker komprimiert werden. Farbräume, die die Intensität (oder Luminanz) als Komponente benutzen sind also zu bevorzugen.

Farbräume (1)

RGB Linear oder nicht linear (R'G'B'). Rot-Grün-Blau. Gut geeignet für HW-Implementierung.

CMY Cyan, Magenta, Yellow (auch CMYK, K = schwarz). Negative Farbmischung (bei Druckern üblich).

XYZ Festlegung durch CIE Kommission 1931. Berechnung aus SPD (spectral power distribution) genau definiert. Heute noch Basis für Definition von Farbräumen.

L*u*v*, L*a*b* Auch von CIE. Perfekte Uniformität. L* ist die Luminanz. Nichtlinear. Aufwendige Berechnung $\Rightarrow$ selten benutzt.

HIS,HSV≈HCV≈HLS,HSB Psychologisch orientiert. Technisch weniger relevant.

- **I**ntensity
- **V**alue: $V = \frac{R+G+B}{3}$
- **H**ue: dominante Farbe (zyklisch angeordnet) $H = 180\frac{0.5(R-G)+(R-B)}{((R-G)^2+(R-B)(G-B))^{1/2}}$, wenn $S = 0$ dann H undefiniert, wenn $B/V > G/V$ dann $H = 360 - H$
- **S**aturation: Sättigung (grau hat am wenigsten Saturation) $S = 1 - \frac{3\min(R,G,B)}{R+G+B}$
- **B**rightness

MTM Nach Munsell. Gute Uniformität. Einfache Transformation von RGB.

Farbräume (2)

YUV Kommt aus NTSC und PAL Video-Standards. Y $\approx$ Luminanz, U und V $\approx$ Chrominanz.

Y'CbCr, Y'PbPr In JPEG und MPEG verwendet. Y'PbPr eher für nicht-digitale Anwendungen.

Y' entspricht ungefähr der Luminanz L* (CIE), sollte eigtl. "Luma" genannt werden. Wenn R', G' und B' nicht lineare 8-bit Werte $(0 \ldots 255)$ sind, dann ist

$$\begin{pmatrix} Y' \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} + \frac{1}{256} \begin{pmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \frac{1}{256} \begin{pmatrix} 298.082 & 0.000 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0.000 \end{pmatrix} \left(\begin{pmatrix} Y' \\ Cb \\ Cr \end{pmatrix} - \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} \right)$$

Y' hat dann einen Wertebereich von $16 \ldots 235$ und Cb und Cr von $16 \ldots 240$.

Wie komprimieren?

Bild- und Videodaten werden meist verlustbehaftet komprimiert. D.h. nach der Wiederherstellung ist das Bild leicht verändert. “Unwichtige” Teile werden einfach unterschlagen. Was unwichtig ist kann über das “human vision system” (**HVS**) modelliert werden.

Um in einem Bild wichtige und unwichtige Information voneinander zu trennen, wird meistens eine frequenz-basierte Transformation angewandt (**transform coding**). Dadurch konzentriert sich die Bildinformation auf tieffrequente Koeffizienten.

Bildkompression besteht daher aus folgenden Schritten:

- Transformation
- Kodieren der Information, welche Koeffizienten vernachlässigt werden
- Kodieren der nicht vernachlässigten Koeffizienten (meist gerundet)
  Dabei wird meist zusätzlich eine verlustfreie Kompressionsmethode angewandt.

Transformationen

In der Bildkompimierung übliche Transformationen sind:

Fouriertransformation Zerlegung des Bildinhalts in $\sin$- und $\cos$-Funktionen bzw. zweidimensionale Produkte davon.

DCT Diskrete Cosinus-Transformation
Ähnlich der Fourier-Transformation aber keine komplexen Koeffizienten.

Wavelettransformation Zerlegung des Bildinhalts in sinusähnliche "Wellchen"
Umgeht das Problem, dass jeder Koeffizient das ganze Bild beeinflussen kann.

Die wichtigsten Unterschiede dieser Transformationen liegen in

- der Fähigkeit zur Informationskonzentration und
- der rechnerischen Komplexität

Fourier Transformation

Hier nur sinnvoll: die diskrete Fouriertransformation (**DFT**)

Wenn $x_0 \ldots x_{N-1}$ eine Folge von reellen Werten ist, dann sind mit

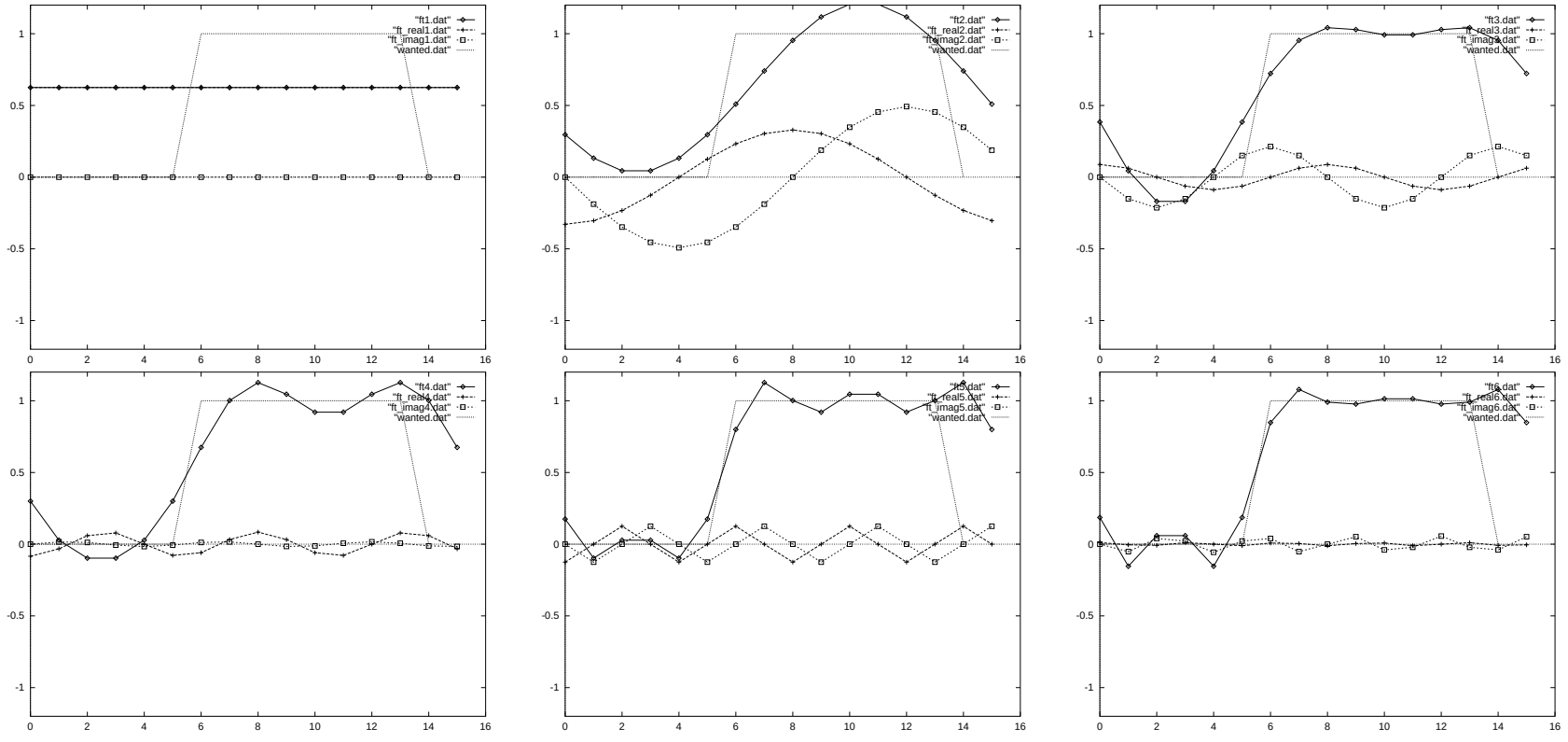
$$\bar{x}_k = \sum_{n=0}^{N-1} x_k e^{-i2\pi kn/N} \quad k \in 0 \ldots N-1$$

$\bar{x}_0 \ldots \bar{x}_{N-1}$ die transformierten Daten. Hier ist zu beachten, dass die $\bar{x}_k$ im allgemeinen komplex sind. Der relle und der imaginäre Anteil gehört jeweils zur Cosinus- bzw. Sinus-Funktion.

Im zweidimensionalen Fall führt man diese Transformation nacheinander auf Zeilen und Spalten durch.

Die Komplexität der Fourier-Transformation ist N^2. Aber es existiert ein schnellerer Algorithmus um diese Koeffizienten zu berechnen: die Fast-Fourier-Transformation (**FFT**). Diese hat eine Komplexität von $N \log(N)$.

FT - Veranschaulichung



Die ersten sechs Koeffizienten der FT und ihr Einfluss auf die rekonstruierte Funktion. Man beachte, dass immer zwei Basisfunktionen dazuaddiert werden ($\cos$ und $\sin$). Die Koeffizienten (zu sehen als Amplitude der Basisfunktionen) werden immer kleiner.

DCT - Diskrete Cosinus-Transformation

ist gegeben durch

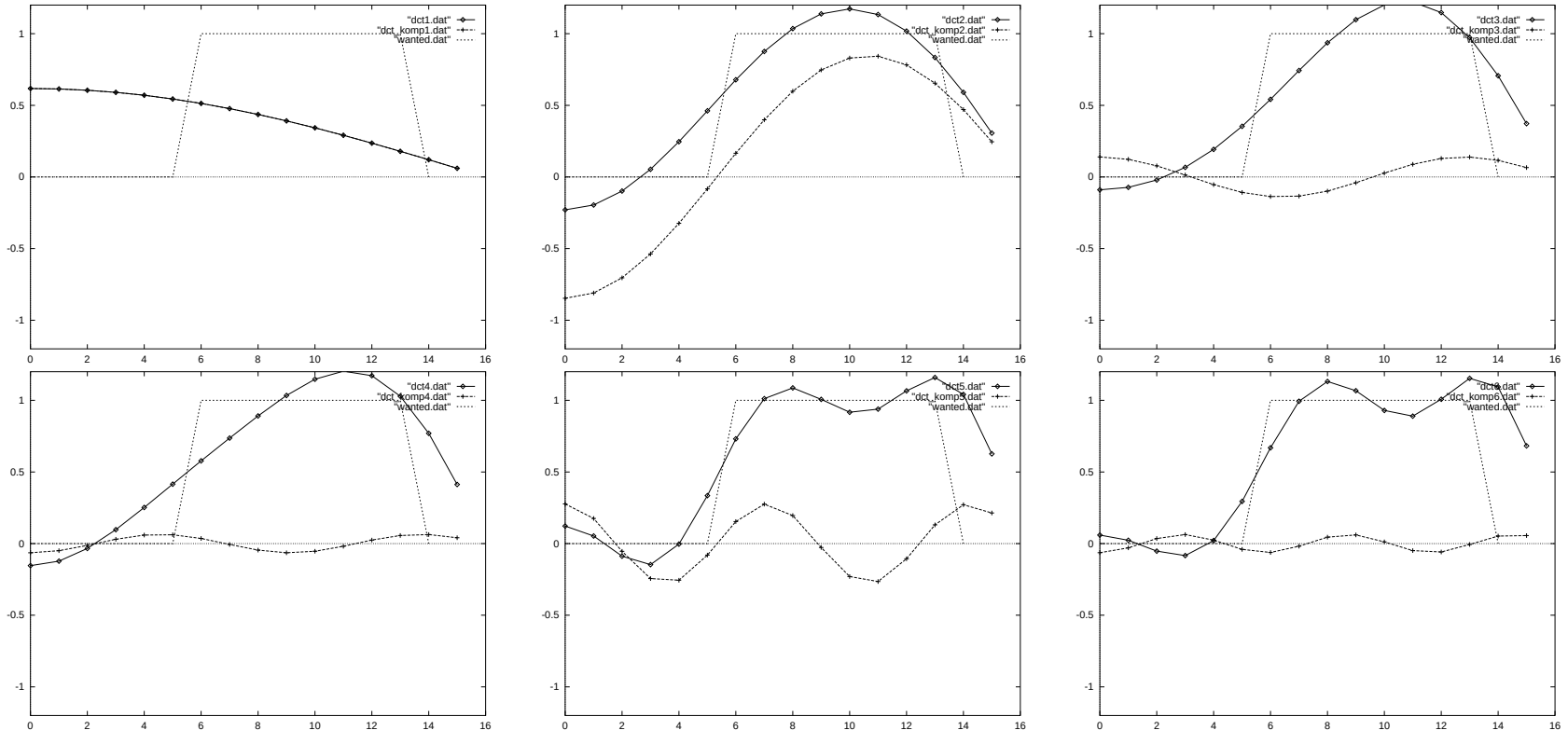
$$\bar{x}_k = c(k) \sum_{n=0}^{N-1} x_k \cos \frac{\pi(2n+1)k}{2N}$$

wobei

$$c(0) = \sqrt{\frac{1}{N}}, \quad c(k) = \sqrt{\frac{2}{N}} \quad k \geq 1\,.$$

Hier sind die Koeffizienten alle reell. Die Komplexität ist die gleiche wie bei der Fourier-Transformation. Auch hier gibt es eine Fast-Version mit der Komplexität $N \log(N)$.

DCT Veranschaulichung



Die ersten sechs Koeffizienten der DCT und ihr Einfluss auf die rekonstruierte Funktion.

Wavelet-Transformation (1D)

Die diskrete Wavelettransformation (**DWT**) basiert auf einer schrittweisen Filterung von Signalen (oder Datenreihen). Filterung ist die Faltung $\otimes$ einer Funktion f mit einem **Filter** h. Im diskreten Fall:

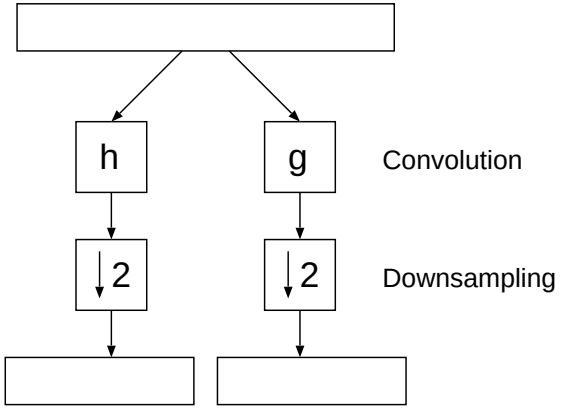
$$(f \otimes h)(t) = \sum_k f(t+k)h(k)$$

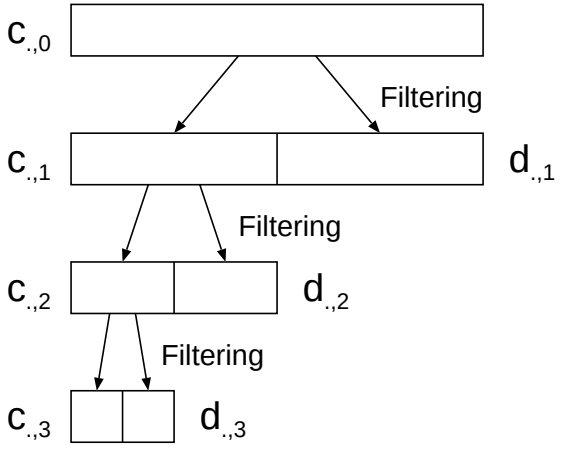
Je nach Länge des Filters h durchläuft k dabei z.B. die Werte $-4 \ldots 5$.

Filter sind in der Lage, gewisse Frequenzbereiche einer Funktion zu unterdrücken. In der Wavelettransformation finden immer zwei Filter (Filterpaar) Anwendung. Ein Hochpassfilter g und ein Tiefpassfilter h. Die Frequenzgänge dieser Filter sollen sich ausschließen, sodass die Daten nach der Filterung in einen tief- und einen hoch-frequenten Teil getrennt ist.

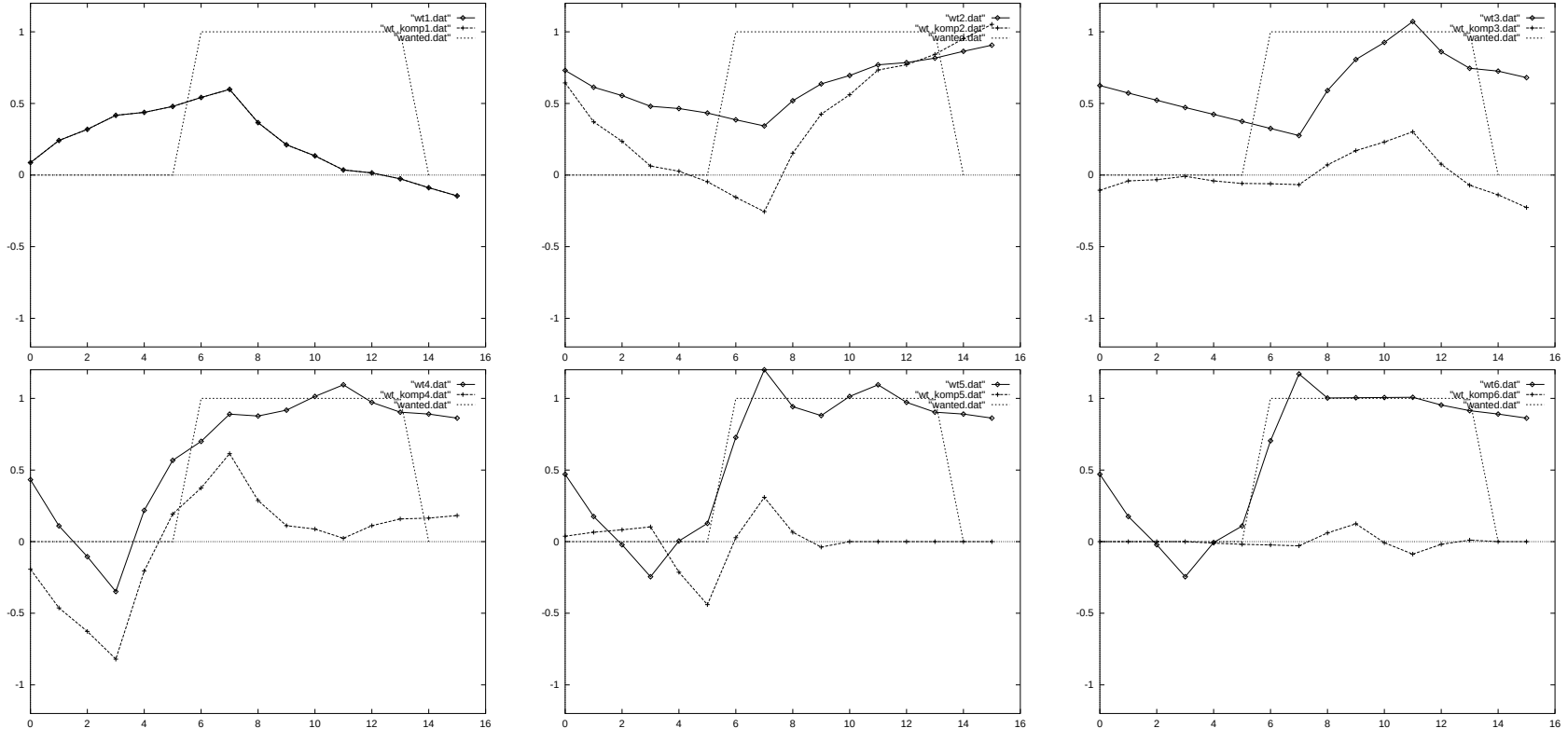
Wavelet-Transformation (1D)

Beim Ergebnis lässt man jeden zweiten Wert aus: **Downsampling**. Dadurch bleibt die Datenmenge gleich. Die gefilterten Datenreihen nennt man **Subbands**. Danach fährt man mit dem tiefpassgefilterten Subband (**approximation subband**, i.Gs. zu **detail subband**) rekursiv fort.





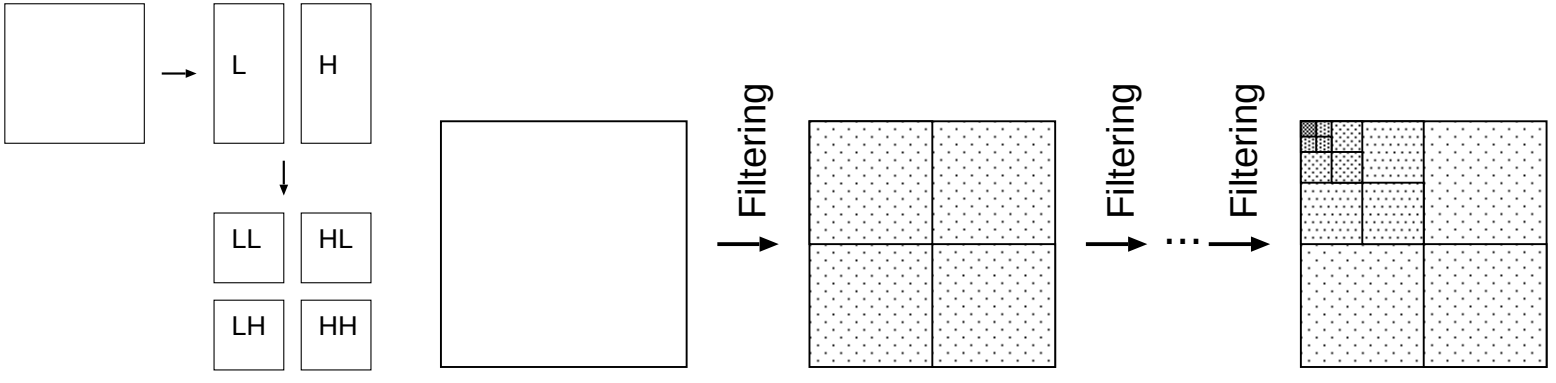
WT Veranschaulichung



Die ersten sechs Koeffizienten der DWT und ihr Einfluss auf die rekonstruierte Funktion. Man beachte, dass vor allem die letzteren (höherfrequenten) Basisfunktionen (*Wavelets*) sich nicht über den ganzen Bereich erstrecken, sondern auch horizontal kleiner werden.

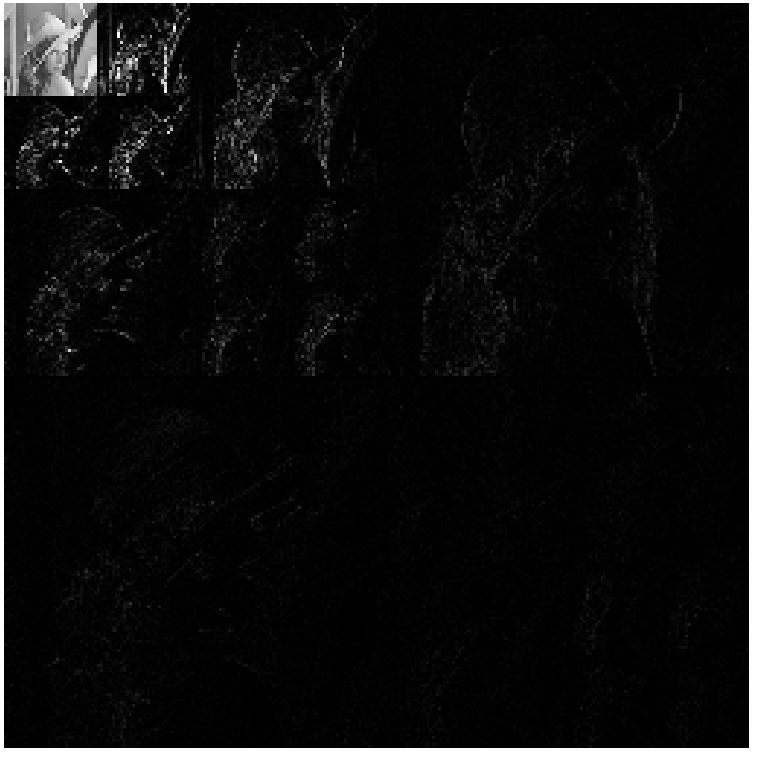
Wavelettransformation (2D)

Bei der zweidimensionalen DWT führt man eine horizontale und eine vertikale Filterung (auf Zeilen und Spalten) hintereinander aus. Dadurch ergeben sich 4 neue Subbands. Davon wird nur eines (LL, das in beide Richtungen tiefpassgefiltert worden ist) weiterzerlegt.



Diese Transformation hat **lineare** Komplexität (N). Die "Energie" des Bildes ist dann in den tieferfrequenten Subbands (links oben) konzentriert.

DWT (2D) Veranschaulichung



Wichtig bei Bildkompression

Bei den Kompressionsverfahren kommt es an auf:

- Gefordertes Laufzeitverhalten (z.B. 30 Bilder pro Sekunde müssen verarbeitbar sein)
- Geforderte Bildqualität. Gemessen mit
  - **MSE** mean squared error

$$\mathrm{MSE}(A, B) = \frac{1}{wh} \sum_{0 \le x < w, 0 \le y < h} (A(x, y) - B(x, y))^2$$

  - **PSNR** peak signal to noise ratio

$$\mathrm{PSNR}(A, B) = 20 \log_{10} \frac{255}{\sqrt{\mathrm{MSE}(A, B)}}$$

- Geforderte Kompressionsrate (z.B. wegen Speicherplatzlimitierung oder Bandbreitenlimitierung bei der Übertragung)
- Gefordertes maximales **Delay**: Verzögerung zw. Sender und Empfänger
- Skalierbarkeit (in Qualität, Auflösung, etc.)

Vorgangsweise bei Bild und Videokompression

- Bei Transformationen mit nichtlinearer Komplexität müssen die Bilder zuerst in Blöcke unterteilt werden, die unabhängig voneinander transformiert werden. (Meist 8×8)
- Dann muss entschieden werden, wieviel Information, die in den Koeffizienten steckt vernachlässigt werden kann/muss. Da man die Kompressionsrate zu diesem Zeitpunkt meist noch nicht richtig abschätzen kann braucht man einen intelligenten Mechanismus, um das zu steuern. $\Rightarrow$ Blockweise kodieren und zwischendurch die Kompression verstärken oder verringern.
- Information über Koeffizienten (Größe, Position, etc.) auf endliche Anzahl von Symbolen abbilden
- Häufige Symbole mit kürzeren Codes kodieren

Quantisierung

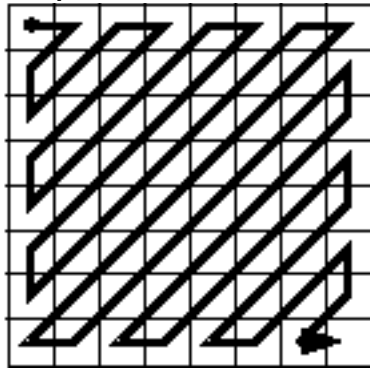
- Die Quantisierung ist meist die einzige Operation, bei der wirklich Information verloren geht.
- Die Koeffizienten die sich aus der jeweiligen Transformation ergeben, sind meistens nicht ganzzahlig (außer bei der ganzzahligen DWT). Bei der Quantisierung werden daraus wieder ganze Zahlen gemacht (Runden).
- Beim einfachen Runden ist die Schrittweite gleich 1. Um andere Schrittweiten zu erreichen dividiert man die Werte vorher einfach durch die Schrittweite. Deshalb wird diese oft Quantisierungsfaktor genannt.
- Bei der verlustfreien Komprimierung (**lossless**) kann es natürlich keine Quantisierung geben.
- Verschiedene Teile der transformierten Daten werden verschieden quantisiert. Entweder auf Koeffizientenbasis (bei DCT-Blöcken) oder subband-weise.
- In gewissen Wertebereichen (z.B. um 0) kann die Quantisierungs-Schrittweite erhöht werden $\Rightarrow$ **Thresholding**
- Eine erweiterte Form der Quantisierung ist die **Vektor-Quantisierung**. Dabei werden mehrdimensionale Vektoren auf vorher festgelegte Vektoren reduziert. Es wird immer der nächstliegende Vektor gewählt, um den Fehler gering zu halten. Die Auswahl der Quantisierungs-Vektoren ist schwierig, die Algorithmen dazu sehr aufwendig und meistens suboptimal.
  Einfaches Beispiel: **Indexed colors**. Im Header einer Bilddatei werden eine Reihe (z.B. 256) von Farben definiert. Für die einzelnen Bildpunkte wird dann nur noch der Index der ähnlichsten Farbe kodiert.

Runlength-Encoding (RLE)

Sehr oft müssen viele gleiche Werte nacheinander kodiert werden. In diesem Fall bietet sich an, den Wert nur einmal zu kodieren, gefolgt von der Anzahl mit der der Wert auftritt.

Meistens handelt es sich dabei um den Wert 0. Es kann sich als gut herausstellen, immer ein Paar (n, w) gemeinsam zu kodieren. Das soll heißen: n Nullen gefolgt vom einmal w. Alle möglichen Paare (n, w) (oder zumindest die häufigsten) werden als einzelnes $Symbol$ kodiert (siehe Entropiekodierung).

Um die Wahrscheinlichkeit von langen Nullfolgen zu erhöhen, müssen die zweidimensional angeordneten Koeffizienten richtig ein eine Reihe gebracht werden. Besser als z.B. zeilen- oder spaltenweise **Scan-Ordnung** eignet sich speziell bei DCT Koeffizienten der **zig-zag scan**:

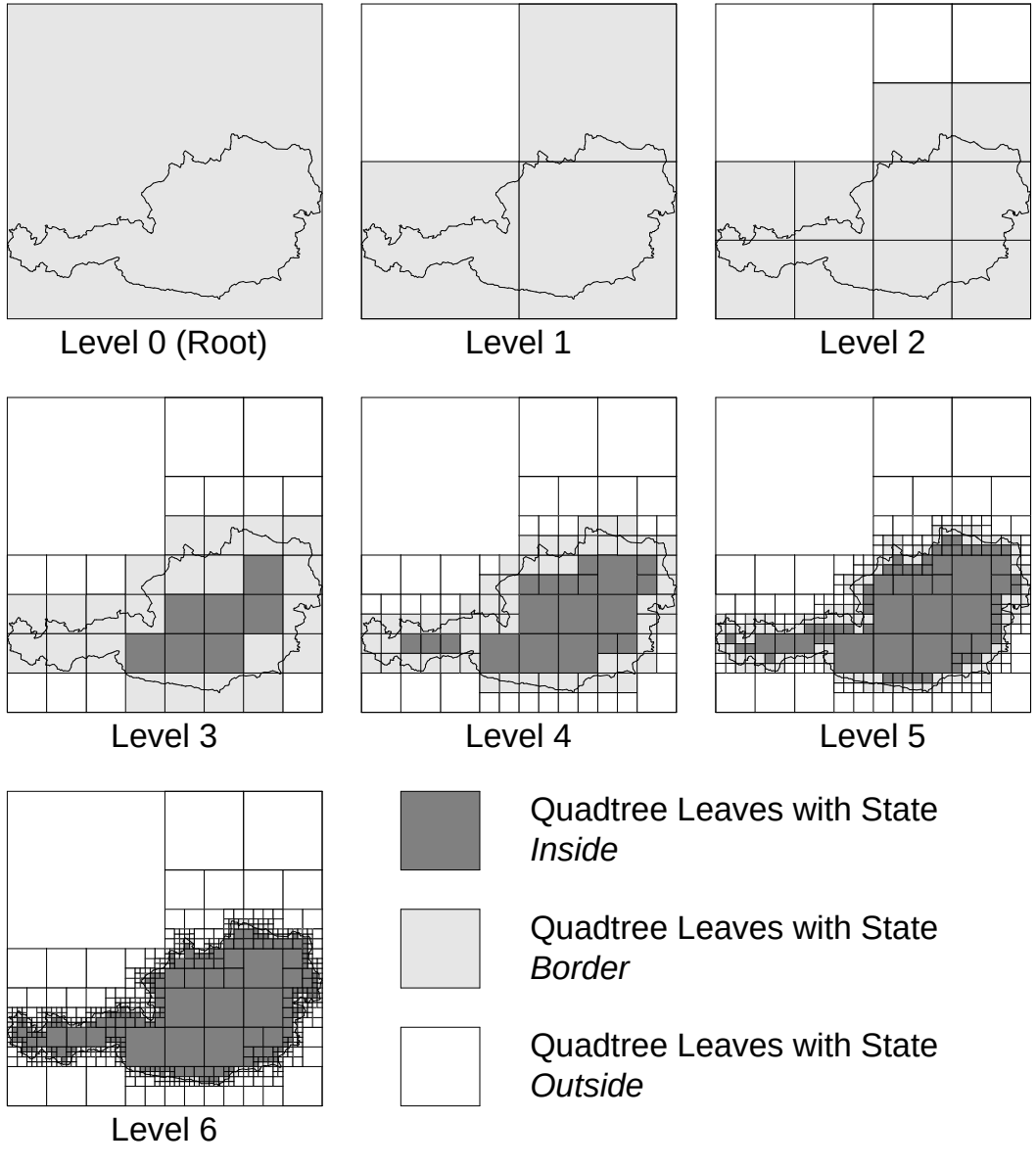


Quad-trees

Die Kodierung der Signifikanz-Information (welche Koeffizienten vernachlässigt werden) läuft auf die Kodierung einer Bitmaske hinaus. Dabei wird das Bild in vier Quadranten unterteilt. Jeder dieser vier Quadranten ist entweder

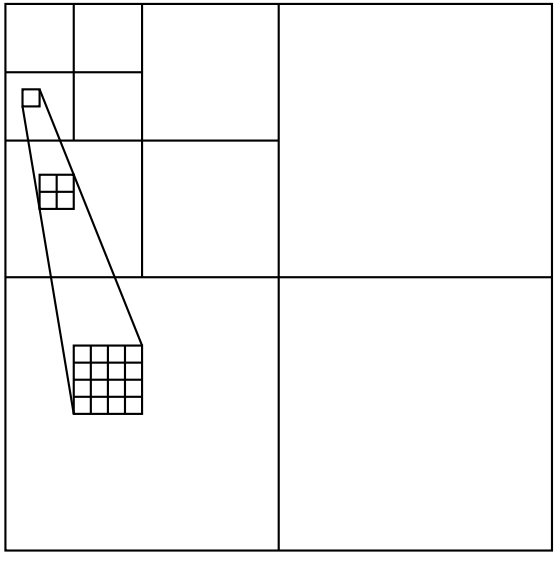
- vollständig innerhalb,
- vollständig außerhalb oder
- teilweise innerhalb und außerhalb

des maskierten Bereichs. Quadranten, die dem letzten Fall entsprechen, werden nach dem gleichen Prinzip rekursiv weiterzerlegt.



Zerotrees

Bei der Wavelet-Transformation gibt es noch einen Spezialfall von Quadtrees: **Zerotrees**. Dabei wird die hierarchische/pyramidale/selbstähnliche Struktur der Subbands ausgenutzt. Koeffizienten mit annähernd gleicher örtlicher Position werden auf allen Detailebenen mit einer Klappe als “nichtsignifikant” kodiert.
Durch die Selbstähnlichkeit der transformierten Daten treten solche Zerotrees recht häufig auf.



Entropiekodierung

Entropiekodierung ist das wichtigste Instrument in der verlustfreien Kompression. Die zu kodierende Information wird unter Verwendung einer Menge von Symbolen dargestellt. Jedes Symbol hat dabei eine gewisse Wahrscheinlichkeit. Ziel ist es nun, Symbole mit hoher Wahrscheinlichkeit mit kürzeren Codes zu versehen. Damit erreicht man die Entropie (Shannon) ziemlich gut. Entropie $\approx$ minimale Datenlänge (Informationsgehalt) für bestimmte Anzahl von Symbolen. Die zwei Hauptvertreter dieser Methode sind:

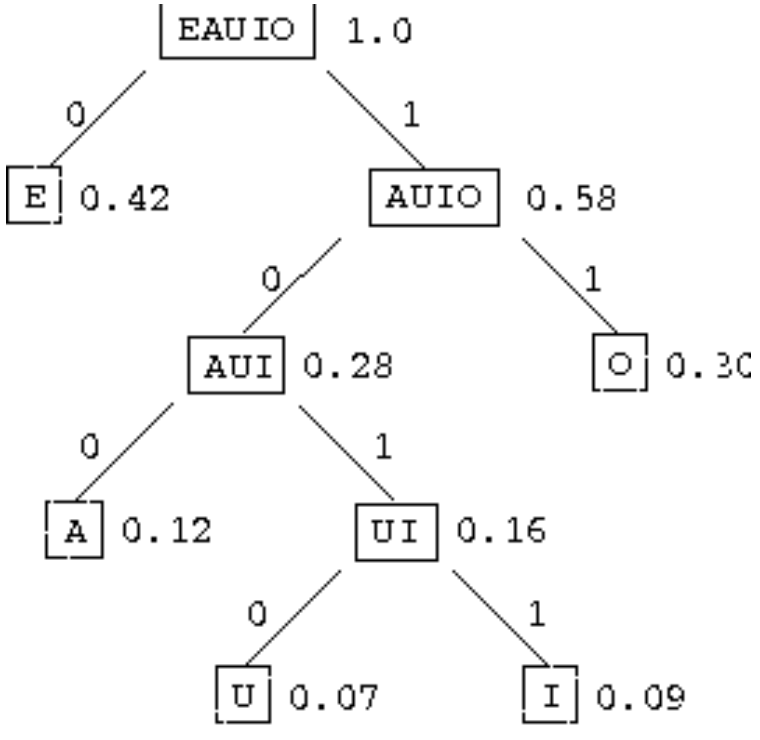
- Huffman-Kodierung
- Arithmetisches Kodieren

Huffman-Kodierung

Jedem Symbol wird ein Code (Folge aus bits) zugeordnet, so dass kein Code ein Prefix eines anderen ist. Dies wird erreicht durch einen binären Baum mit den Symbolen an den Blättern. Verzweigungen nach links mit 0, nach rechts mit 1 kodieren. Unwahrscheinliche Symbole werden weiter vom Urpsrung entfernt angeordnet.

Die (bitweise) in einen **Bitstream** hintereinandergeschriebenen Codes können dann eindeutig aus diesem wieder rekonstruiert werden.

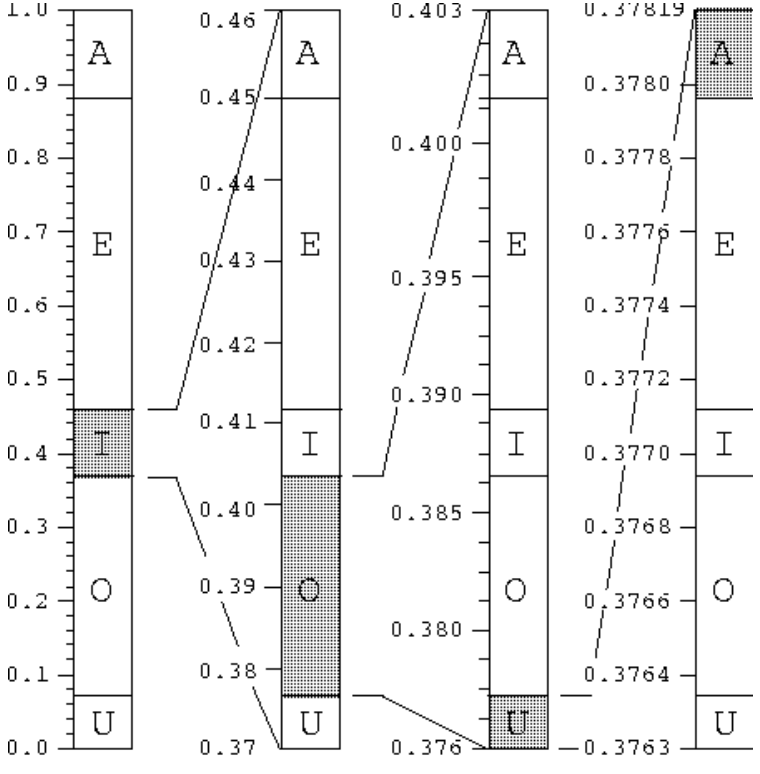
Nachteil: Bei z.B. nur zwei Symbolen muss man von einer 50:50 Wahrscheinlichkeitsverteilung ausgehen.



Arithmetisches Kodieren

Zur Vorstellung dient hier das Einheitsintervall $[0, 1)$. Dieses Intervall wird in Teilintervalle unterteilt. Für jedes Symbol gibt es ein Teilintervall. Die Länge des Teilintervalls entspricht der Symbolwahrscheinlichkeit.

Wird nun ein bestimmtes Symbol kodiert, so wird das zugehörige Intervall ausgewählt. Mit dem nächsten Symbol verfährt man nun gleich, nur dass man das zuvor ausgewählte Intervall verwendet. So werden die Intervalle immer kleiner und konvergieren zu einem bestimmten Punkt am Intervall. Die binären Nachkommastellen derjenigen Zahl, die in dem resultierenden Intervall nach der Kodierung des letzten Symbols liegt, stellen den Bitstream dar. In diesem Fall 0.3780 (binär halt).



Predictive Coding

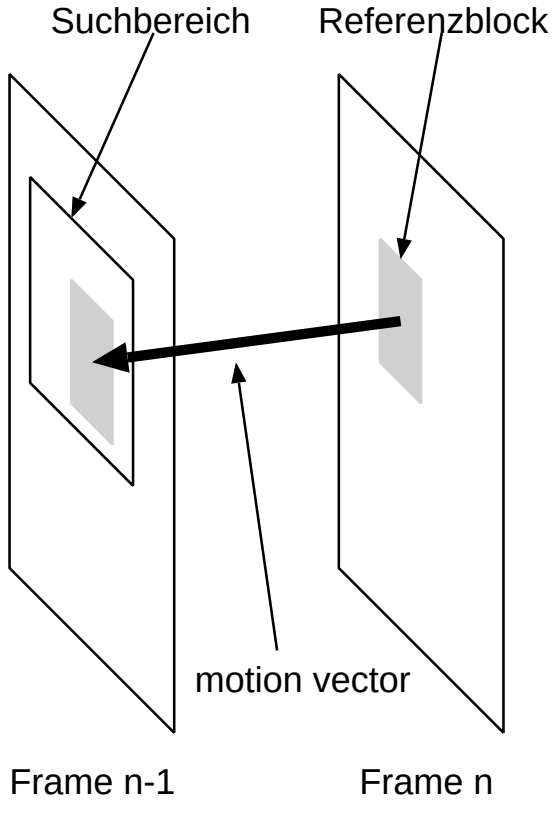
Predictive Coding wird meist im Bildbereich angewendet. Dabei wird der Wert eines Pixels durch benachbarte (bereits kodierte) Pixel geschätzt (vorausgesagt). Die Schätzung wird von dem wirklichen Wert subtrahiert. Das zurückbleibende Bild heißt **Residuum** oder **Residual**. Es beinhaltet weniger "Energie" und ist leichter zu komprimieren.

Beim *linearen* Predictive Coding ist die Schätzung eine Linearkombination der Nachbarpixel. Die optimalen Koeffizienten dieser Linearkombination für ein ganzes Bild können berechnet werden. Es gibt auch nichtlineares Predictive Coding.

Motion Estimation/Compensation

Als besondere Form von Predictive Coding kann man Motion Estimation (**ME**) und Motion Compensation (**MC**) ansehen. Hier nur die wichtigste Variante: **Block matching**.

Frame wird in Blöcke unterteilt. Zu jedem Block wird im vorhergehenden Frame nach einem Block (pixelweise verschoben) gesucht, der ähnlich ausschaut (aufwendig). Der Vektor, um den der Block verschoben wird, heißt **motion vector**. Dann wird der Block subtrahiert und das Residuum kodiert.
Es gibt auch MC mit Bezug auf zeitlich danach kommende Frames (die natürlich vorher kodiert werden müssen). Und bidirektionale MC.



Formate

Bildformate

JPEG wohl jedem bekannt

JPEG2000 neuer Standard mit vielen neuen Features

SPIHT akademisch. komprimiert sehr gut. basiert auf WT und Zerotrees. sehr schnell.

Videoformate

MPEG MPEG-1 und MPEG-2. Entwickelt für digitales TV. MPEG-3 war usprünglich für HDTV gedacht, dann tats eine Erweiterung von MPEG-2 aber auch.

H.263 wie MPEG nur für Videoconferencing. Nur Video, kein Audio.

H.32x mit Audio für ISDN, POTS und IP

Multimediaformate

MPEG-4 Zerlegt Videos in Bestandteile und setzt sie wieder zusammen.

MPEG-7 Fügt alle möglichen Informationen dazu für den semantischen Zugriff.

JPEG

- Bevorzugter Farbraum: YCbCr
- Chroma downsampling: meistens 2:1 horizontal und oft sogar 2:1 vertikal
- Das Bild (genauer: jede Farb-Komponente) wird in 8×8-Pixel-Blöcke unterteilt. Diese werden DCT-transformiert.
- Quantisierung auf Koeffizientenbasis. $\Rightarrow 8 \times 8$ Quantisierungs-"Matrix".
- Die 8×8 quantisierten Koeffizienten werden run-length- und entropiekodiert.
- Zigzag-Scan.
- Es werden folgende Symbole kodiert:
  - Länge der Zero-Runs zusammen mit Bitlänge des nachfolgenden Koeffizienten
  - Wert des Koeffizienten (mit obiger Bitlänge)
  - Blockende (wenn alle restlichen Koeffizienten 0 sind)

JPEG

- Es ist Huffman-Kodierung aber auch arithmetische Kodierung möglich (ca. 5-10% bessere Kompressionsleistung)
- DC-Koeffizienten (der ganz links oben) werden (predictive!) als Differenz zum vorigen Block kodiert.
- Kompressionsrate über Quantisierungsmatrix einstellbar. Meistens eine best. Matrix linear skaliert.
- Im Header einer JPEG-Datei befinden sich (neben Bilddimensionen) hauptsächlich die Quantisierungsmatrix und die Huffman-Tabellen.

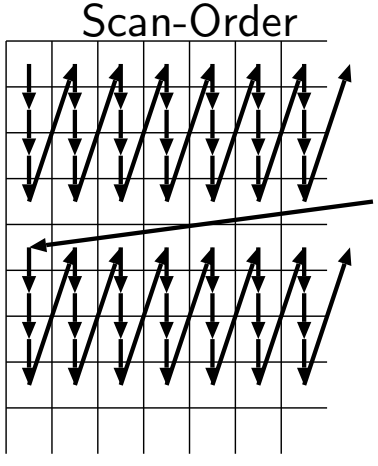
JPEG - Erweiterungen

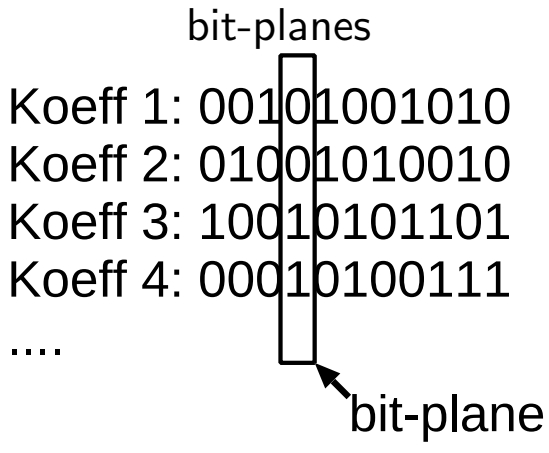
- Dekodierer kann noch: Smoothing wegen Blocking-Artefakte
- Lossless mode:
  - Keine DCT-Transformation
  - Predictive-Coding: jedes Pixel durch Pixel links und oberhalb voraussagen
  - Es stehen 8 verschiedene Prediction-Funktionen zur Verfügung
  - Danach: Zigzag-Runlength-Huffman wie üblich
- Variable Quantisierungsmatrix in versch. Bildbereichen (**region of interest**).
- Progressive mode:
  - Aufteilen in mehrere Scans
  - Erster Scan in Low-Quality
  - Nachfolgende Scans kodieren die Differenz zwischen vorhergehenden Scans und Originalbild.
  - Kein nennenswerter Kompressionsverlust.
  - Kombination mit lossless: Letzter Scan lossless
- Hierarchical mode: wie progressive. Unterschied: vorhergehende Scans sind horiz. und/oder vert. downgesampled.

JPEG2000

- Verwendet die Wavelet-Transformation
- Filter: für verlustbehaftete Kompression Daubechies-Biorthogonal-7/9, für lossless Integer-3/5. (Die Ziffern benennen die Länge des Hoch- bzw. Tiefpassfilters)
- Das Bild wird in sog. **tiles** (große Blöcke) unterteilt.
- Die Tiles werden 2-D wavelet-transformiert
- Die Subbands werden in sog. **code-blocks** unterteilt (Seitenlänge muss 2-er Potenz sein)
- Die code-blocks werden unabhängig voneinander kodiert.
- Quantisierung auf code-block-Basis.
- Die bitstreams, die aus den code-blocks entstehen, können in packets unterteilt werden, wobei aufeinanderfolgende packets die Bildqualität sukzessive verbessern.
- Die packets aus verschiedenen tiles, components, sub-bands und code-blocks sollen nun so angeornet werden, dass bei Abschneiden der Datei ab einem bestimmten Punkt (Verlust der nachfolgenden Packets) die Bildqualität in Anbetracht der neuen Kompressionsrate optimal ist. Das ist eine schwierige Aufgabe.
- Über die Anordnung der Packets kann bestimmt werden, ob zuerst eine geringere Auflösung übertragen werden soll (resolution pregressive) oder zuerst schlechtere Qualität (layer progressive) usw. (component progressive, . . .)
- Region-of-interest durch Anpassung der Quantisierung in code-blocks.

JPEG2000 - Wie entstehen die Bits?





- Höherwertige Bitplanes zuerst kodieren
- Arithmetisches Kodieren der Bits aufgrund von Nachbar-Koeffizienten (**context-based**)
- Unterscheidung: significance (führende 1-er), refinement (dahinterliegende bits), sign (Vorzeichen)
- Drei Passes:

  significance + sign coding (für jede Kombination aus Nachbarschafts-Signifikanz eine eigene Wahrscheinlichkeitsverteilung mitführen.

  refinement coding 50:50%-Wahrscheinlichkeitsverteilung

  cleanup + sign coding Sehr unwahrscheinliche Koeffizienten am Schluss mittels run-length coding.

MPEG

- MPEG = Moving Picture Expert Group
- Geschichte: MPEG-1 1991, MPEG-2 1994 (inkludiert interlaced TV, erhöhte target bit-rate)
- Layers:
  - Video Sequence
  - GOP (Group Of Pictures): zusammengehörige, gemeinsam kodierte Frames
  - Picture (auch Frame)
  - Slice
  - Macroblocks 16×16 Luminanz + 2 * 8×8 Chrominanz (YCbCr)
- Chrominanz-Downsampling (eigenwillige Notation):

  4:4:4 nicht downgesampled
  4:2:2 horizontal 1:2
  4:2:0 horizontal und vertikal 1:2

MPEG

- GOP:

  I-Frame kodiert wie JPEG

  Skalierung der Q-Matrix auf Makroblock-Ebene zwecks **bit-rate control**

  P-Frame Motion compensated mit block matching. Nur forward prediction aus I- oder P-frames.

  Bsp: IPPPPIPPPPIPPPP...

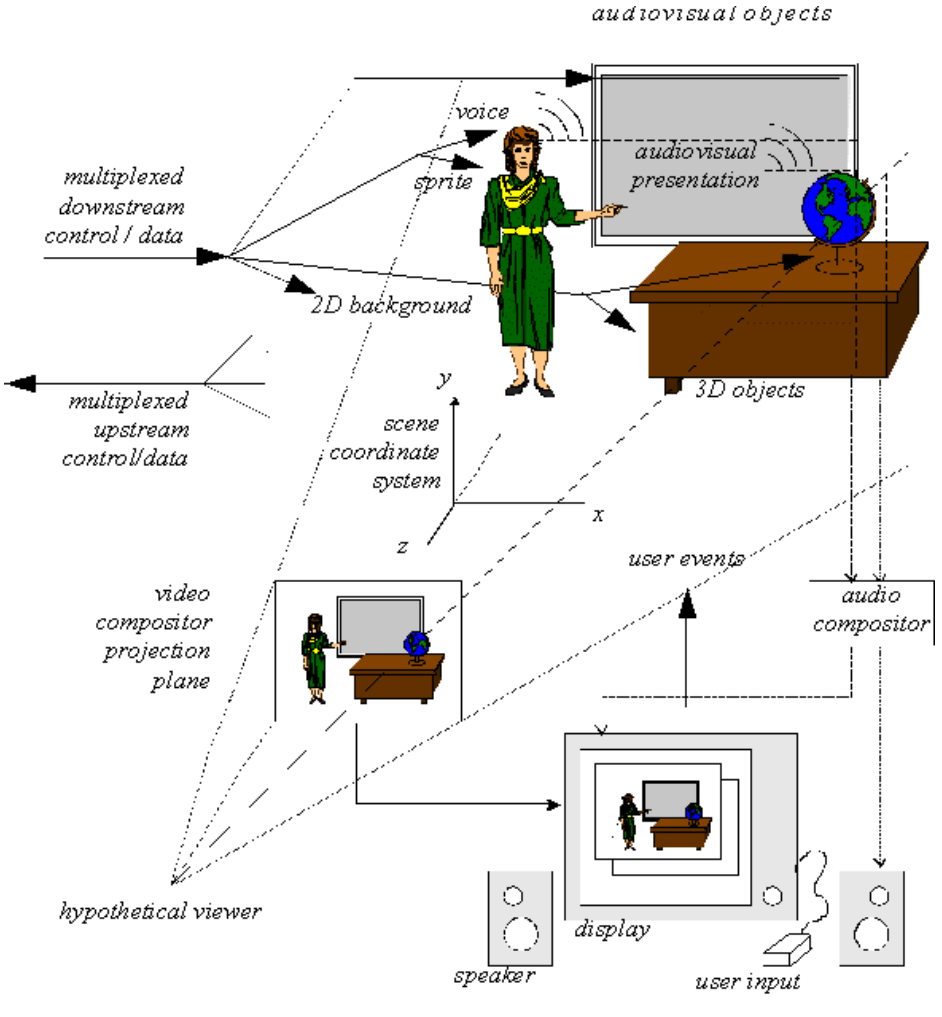
  B-Frame forward und backward predicted (bi-directional) (erhöht Delay)

  üblich: IBBPBBPBBPBBPBBIBBPBBP...
- Chrominance motion vectors werden von der Luminanz übernommen
- Für predicted blocks ist die Quantisierungsmatrix konstant
- Wie motion estimation genau durchgeführt wird ist nicht vorgeschrieben. $\Rightarrow$ rate-performance tradeoff
- Möglichkeit: Macroblocks ohne vernünftigem Match normal kodieren

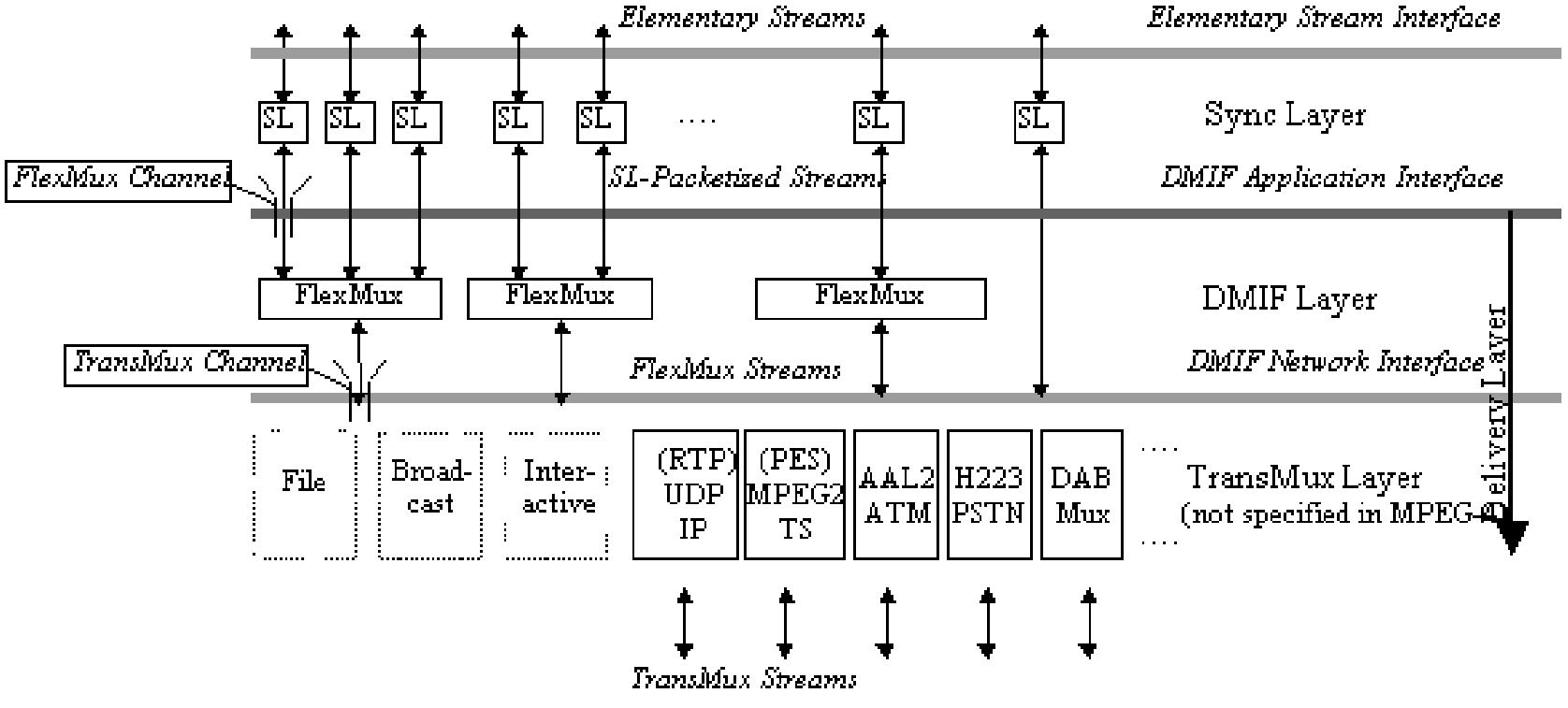
MPEG-4

- 1999 abgeschlossen
- Mehr Anwendungen:
  - Digital television,
  - Interactive graphics applications (synthetic content),
  - Interactive multimedia (World Wide Web, distribution of and access to content)
- Aufgaben von MPEG-4
  - Kodierung von "media objects" (Video, Audio, Stillbild (z.B. als Hintergrund), synthetisch, Text, ...)
  - Beschreibung der "composition" dieser media objects zu einem Ganzen
  - Multiplexen der kodierten Objekte (Synchronisierung, QoS, . . .)
  - Interaktion des Empfängers mit der audiovisuellen Darstellung (z.B. Verschieben des Nachrichtensprechers)
- Skalierbarkeit in Qualität, Auflösung (räuml. und zeitl.), Komplexität muss gegeben sein.
- VOP (video object planes) = "Ausschneiden" von Objekten mittels Shapes (Bitmaske) oder Alpha-Channel (mit Transparenz). Nur das Objekt selbst wird kodiert.
- DMIF (Delivery Multimedia Integration Framework) steuert den Zugriff auf Multimedia-Daten. Ähnlich ftp, aber unabhängig davon, ob man auf ein Netzwerk, ein TV-Kabel oder einen Datenträger zugreift.

MPEG-4 media objects



MPEG-4 Multiplexing im DMIF



MPEG-7

- noch in Arbeit
- Ziel: Semantische Inhaltsbeschreibung
- Scene cuts
- Author-, Copyright und Preisinformationen
- Links
- Suche nach bestimmten Dingen (z.B. bellender Hund) in Multimediadaten
- Kontext (z.B. "Bericht über olympische Spiele 1996, Hürdenlauf, Männer")
- Hauptelemente

  D Descriptors: Repräsentation der Inhalte

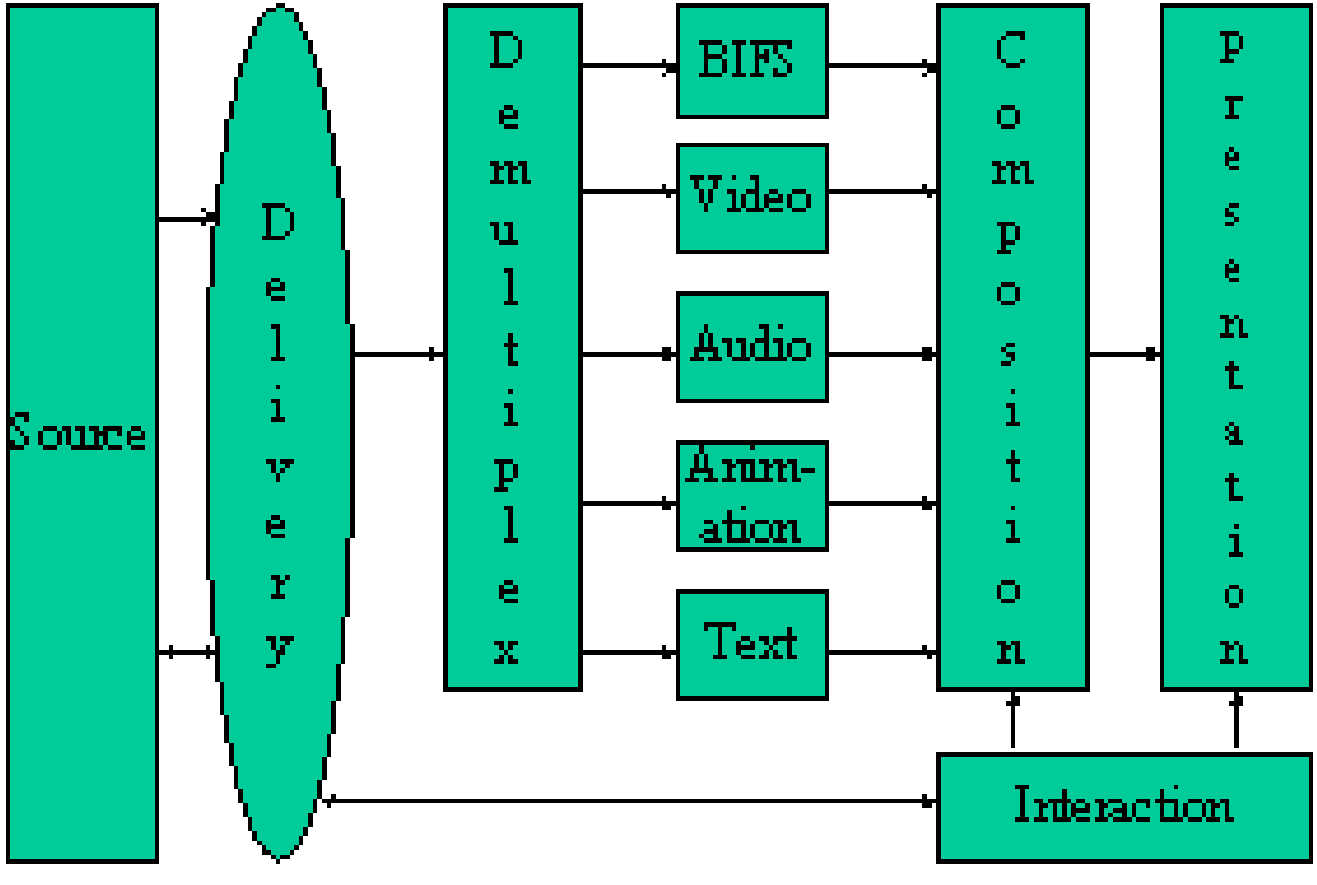
  DS Description Schemes: Zusammenfassung mehrerer Descriptors (oder DS) und Beschreibung ihrer Beziehung

  DDL Description Definition Language: Erlaubt die Erzeugung neuer DS

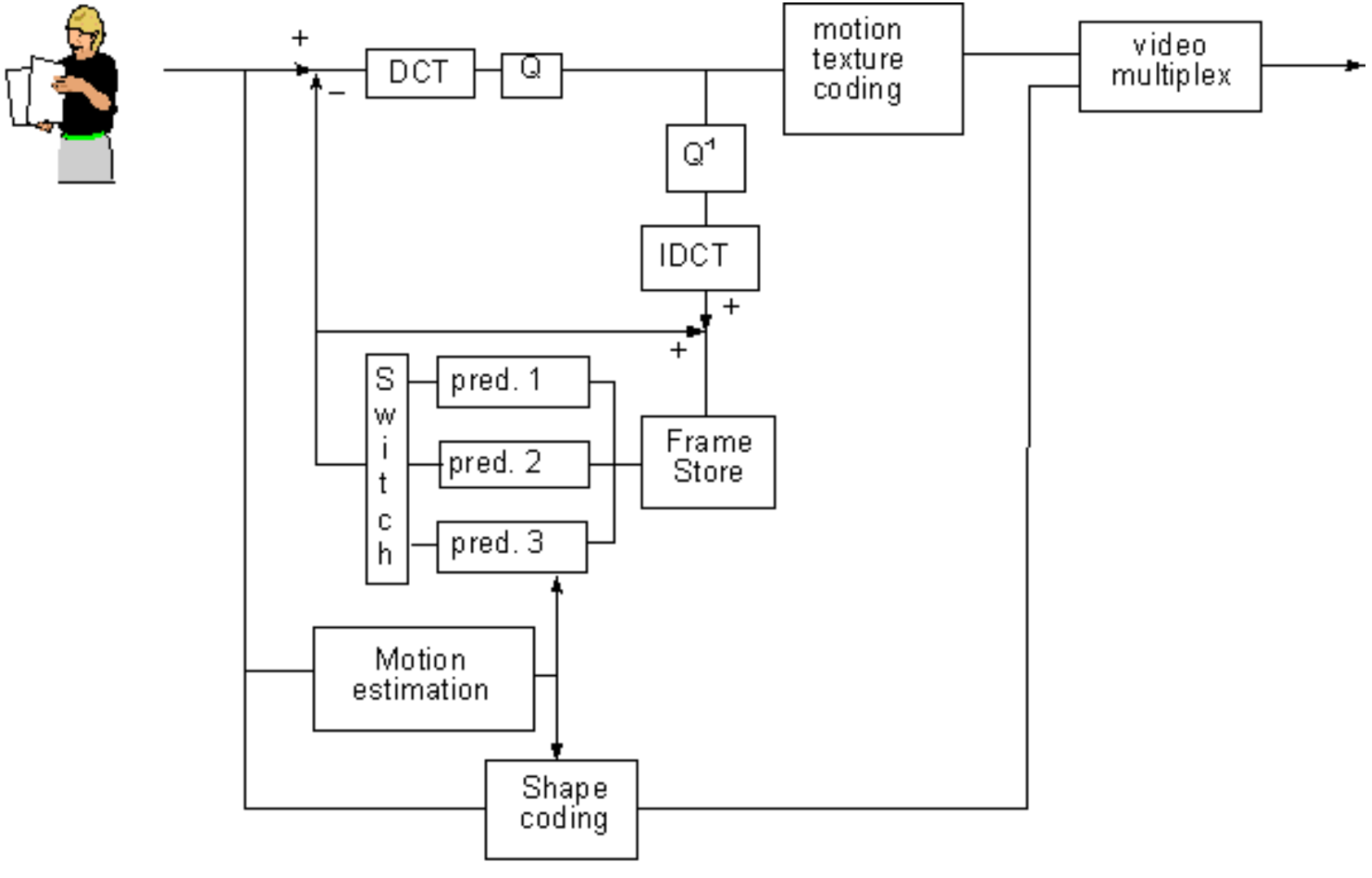
MPEG-4 Anforderungen

- Multimedia-Material ist: **natürlich**, **synthetisch** oder beides
- Präsentation der Multimediadaten vom Benutzer zu beeinflussen
- Interaktion (client-side und server-side)
- Datenfluss **von** und **zu** verschiedenen Quellen
- Transparenter Zugriff auf den Kommunikationskanal (DMIF)
- Übertragungsraten von sehr niedrig bis sehr hoch
- Skalierbarkeit wegen variabler Kapazitäten der Kommunikationskanäle, Endgeräte, Speichermedien
- Verarbeitung in **real time** oder **non real time**
- Widerstandsfähigkeit gegenüber Fehler (**error resilience**)

MPEG-4 Kommunikationsstruktur



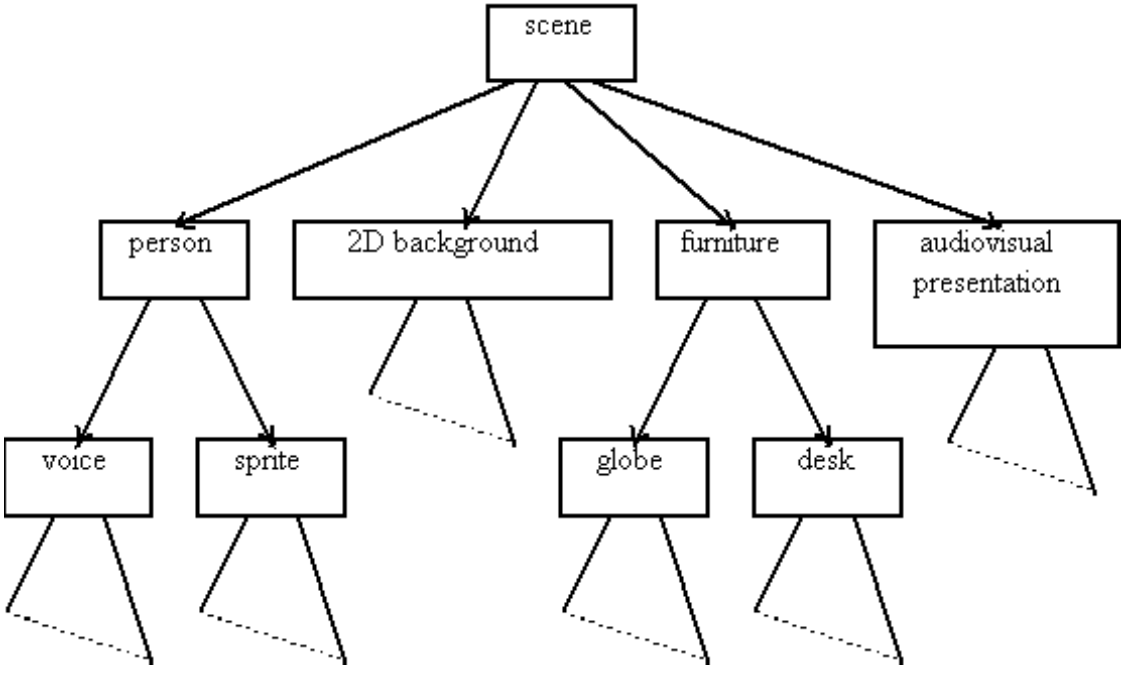
MPEG-4 Video Coding



MPEG-4 Video Coding (2)

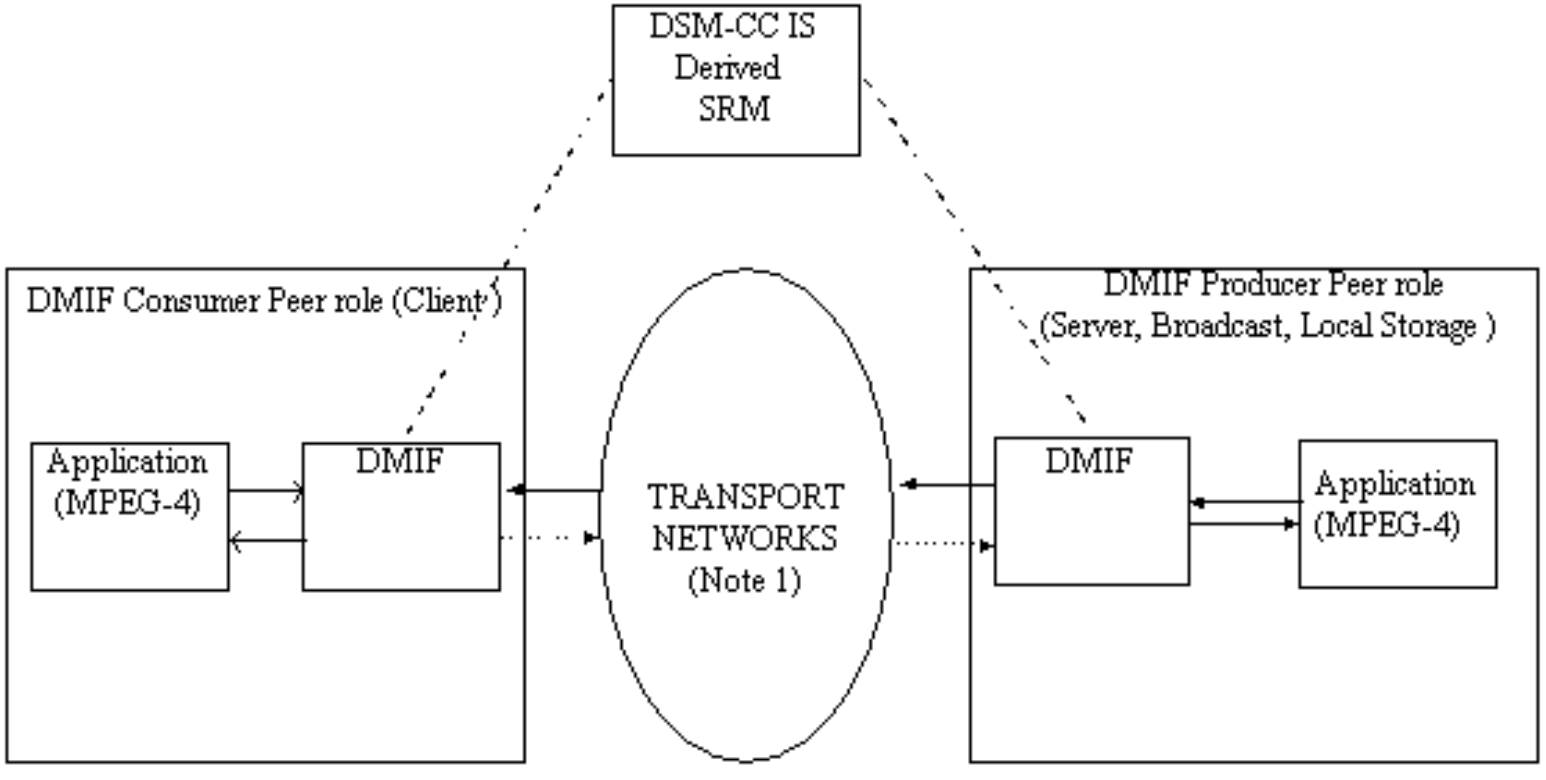
- Standard 8x8 oder 16x16 Pixel Block-based Motion-Estimation und Compensation
- **Global Motion Compensation** mit 8 Motion-Parametern für affine Transformation (**Kamerabewegung**)
- Global Motion Compensation basierend auf einem statischen **sprite** = großes fixes Bild (Hintergrund), das affin transformiert wird
- Global Motion Compensation mit dynamischen sprites, die während der Szene aufgebaut werden.
- Skalierbarkeit bezüglich Komplexität, Qualität, räumlicher Auflösung, zeitlicher Aufösung
- Widerstandsfähigkeit gegenüber Fehler (**error resilience**) durch:
  - **Resynchronisation** durch regelmäßiges Einfügen von Markern in den Bitstream
  - **Data Recovery** durch **error correcting codes**, **reversible variable length codes** (Huffman Codes, die auch von hinten nach vorn gelesen eindeutig sind), etc.
  - **Error Concealment** Fehler verbergen: z.B. einfach Block von vorhergehendem Frame kopieren, wenn ein Fehler erkannt wurde. Stark mit Motion Compensation verknüpft.

MPEG-4 Scene Description



- stark angelehnt an die **virtual reality modelling language** (VRML)
- Jedes Objekt (auch die Szene) hat ein eigenes Koordinatensystem
- Jedes Objekt hat eine Position und Orientierung im übergeordneten Koordinatensystem
- Manche Objekte lassen sich parametrisieren (z.B. synthetischer Kopf)

MPEG-4 Delivery Multimedia Integration Framework (DMIF)

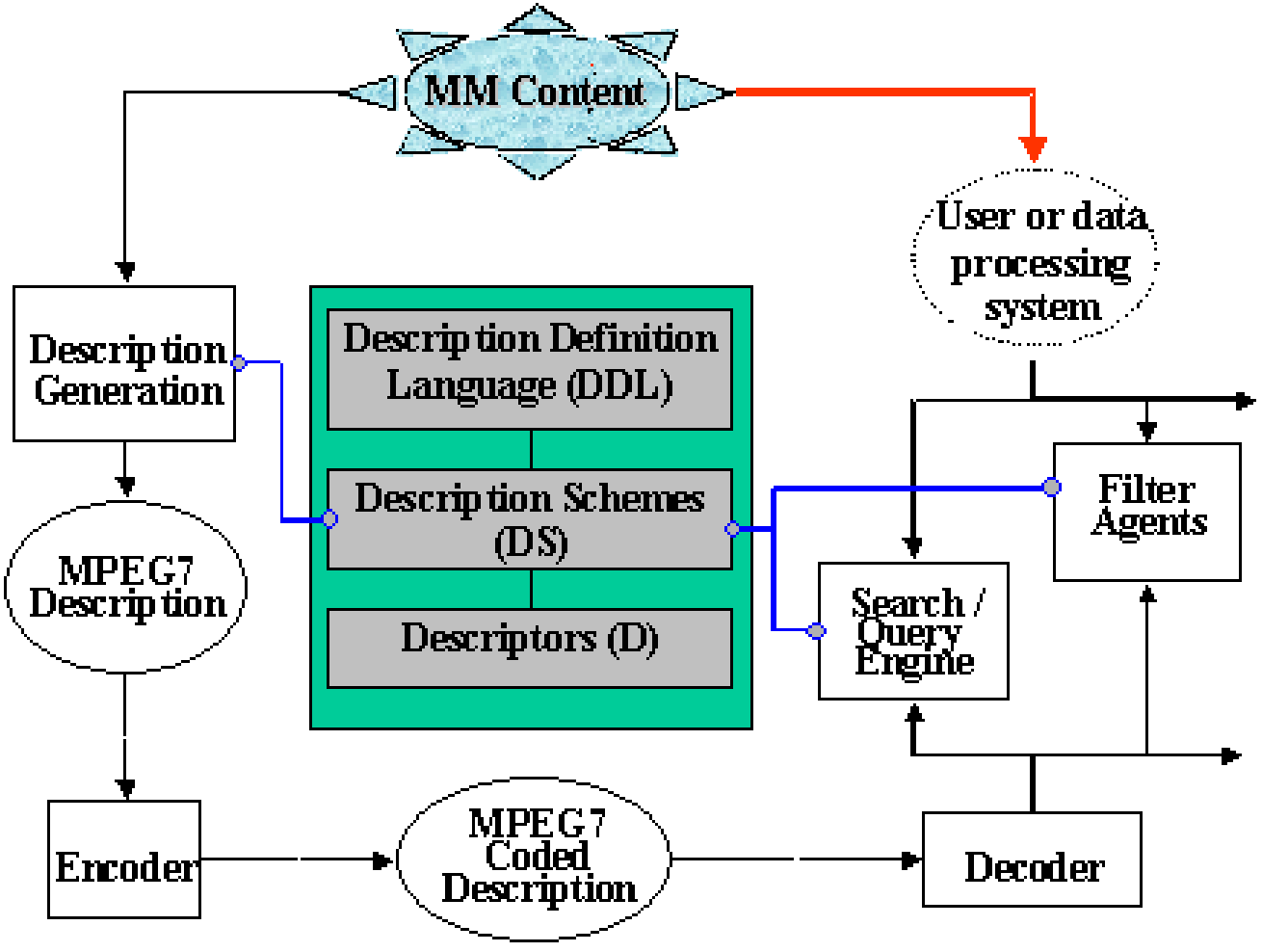


..... = Not present in case of pure broadcast SRM= Session and Resource Management function

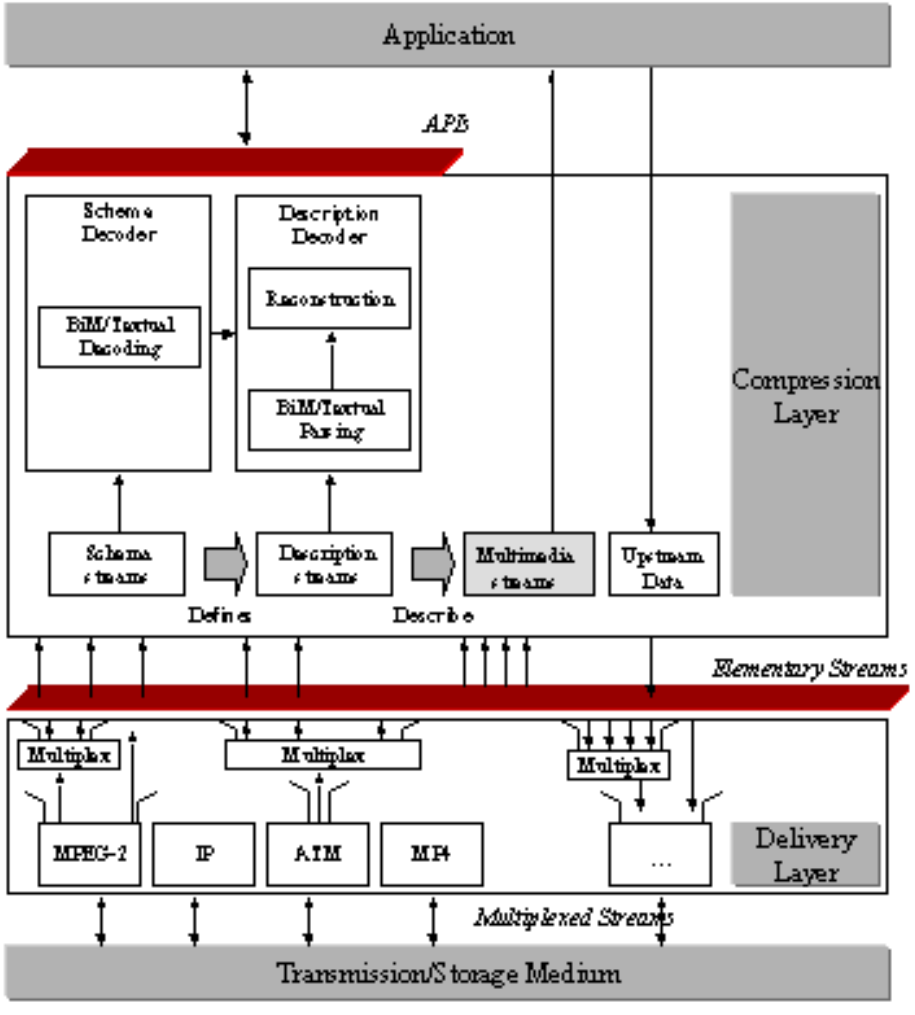
- - - - - = Invoked on demand

Note 1: Includes I/O bus and drivers for DVD in case of local terminal storage

MPEG-7 Anwendung



MPEG-7 Struktur



- Gemultiplexte Objekte und Daten können über alle möglichen Transportschichten (Netzwerk, MPEG-2, etc.) übertragen werden
- Scheme/Description Streams können textuell (DDL) oder binär (BiM) kodiert werden

MPEG-7 DDL, DS und D

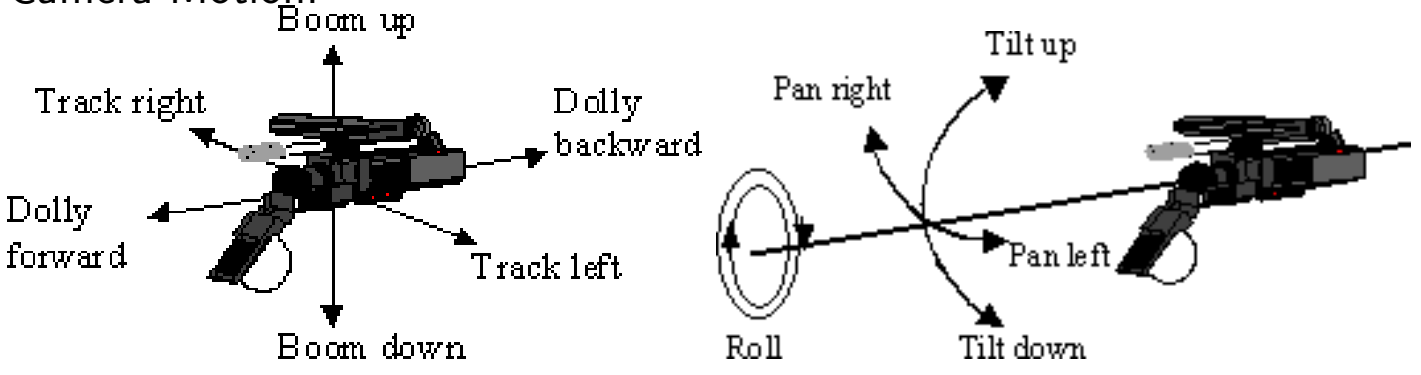
- Descriptor (D)
  - Beschreibt ein Multimediaobjekt in gewisser Weise
  - Meist low-level features (meist automatisch generiert)
  - Aber auch high-level (Semantik, Emotion, ...) (meist nicht automatisch generiert)
- Description Schema (DS)
  - Strukturierte Kombination aus Ds und DSs
  - Wichtigste DS im Standard vorgesehen
  - Neue DS frei definierbar
- Description Definition Language (DDL)
  - Dient der Formulierung von DSs
  - Stark angelehnt an XML (eXtensible Markup Language)
  - Erweitert um: Array-, Matrix- und Zeit-Datentypen

MPEG-7 Visual Description Tools

- Grundstrukturen: Grids, Koordinaten, Views, Zeitreihen, Interpolationsfunktionen
- Farbe:
  - Farbraum, Farbquantisierung
  - Dominant Color(s): Häufigste Farbe in einem best. Bereich
  - Scalable Color: Farb-Histogramm mittels Haar-Wavelet kodiert
  - Color Structure: Analyse über Beziehung zu Farben der Nachbar-Pixel
  - Color Layout: räumliche Farbverteilung
  - GoF/GoP Color: Scalable Color für Bildsequenzen gemittelt über $average$, $median$ oder $intersection$
- Textur:
  - Homogenous Texture: Energieverteilung nach Gaborfilterung (62 * 8 bits)
  - Texture Browsing: nur 12 bits, ähnlich wie homogenous, entspricht menschlichen Wahrnehmung: Hauptorientierung, Regularität, Grobheit
  - Edge Histogram: Verteilung von 5 verschiedenen Typen von Kanten
- Shape:
  - Region Based: mittels bitmap
  - Contour Based: Umrandung als Linienzug
  - 3D-Shape: 3-dimensionale Objekte mittels Polygon-Meshes

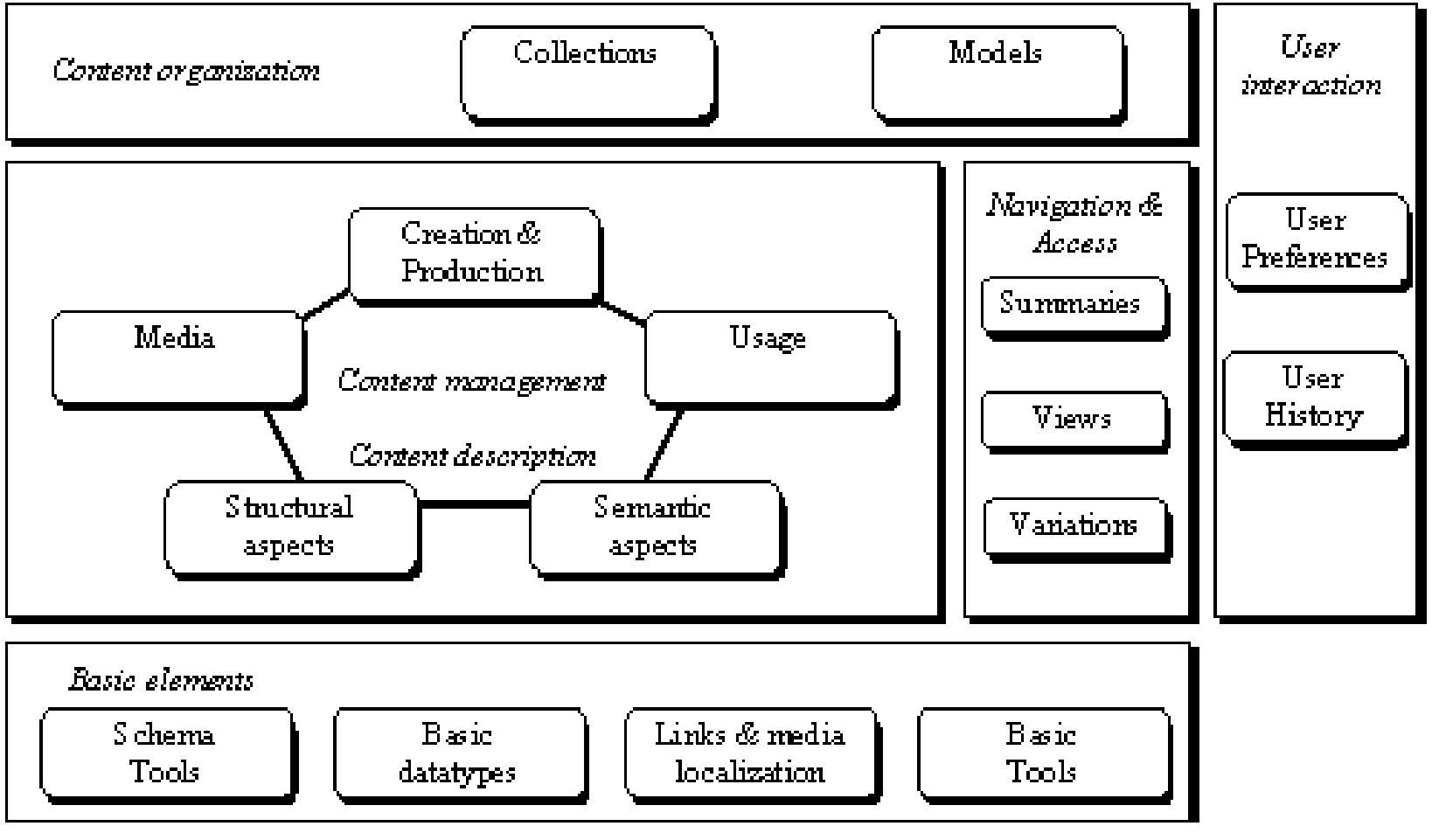
MPEG-7 Visual Description Tools

- Motion:
  - Camera Motion:



  - Motion Trajectory: Bewegung eines Objekts
  - Parametric Motion: inkludiert Rotation, Deformation, u.s.w.
  - Motion Activity: Viel oder wenig Bewegung
- Sonstiges:
  - Localization: Auffinden von Regionen in Bildern und Sequenzen
  - Gesichtserkennung: “Normalisiertes Gesicht” in einem kleinen Raster

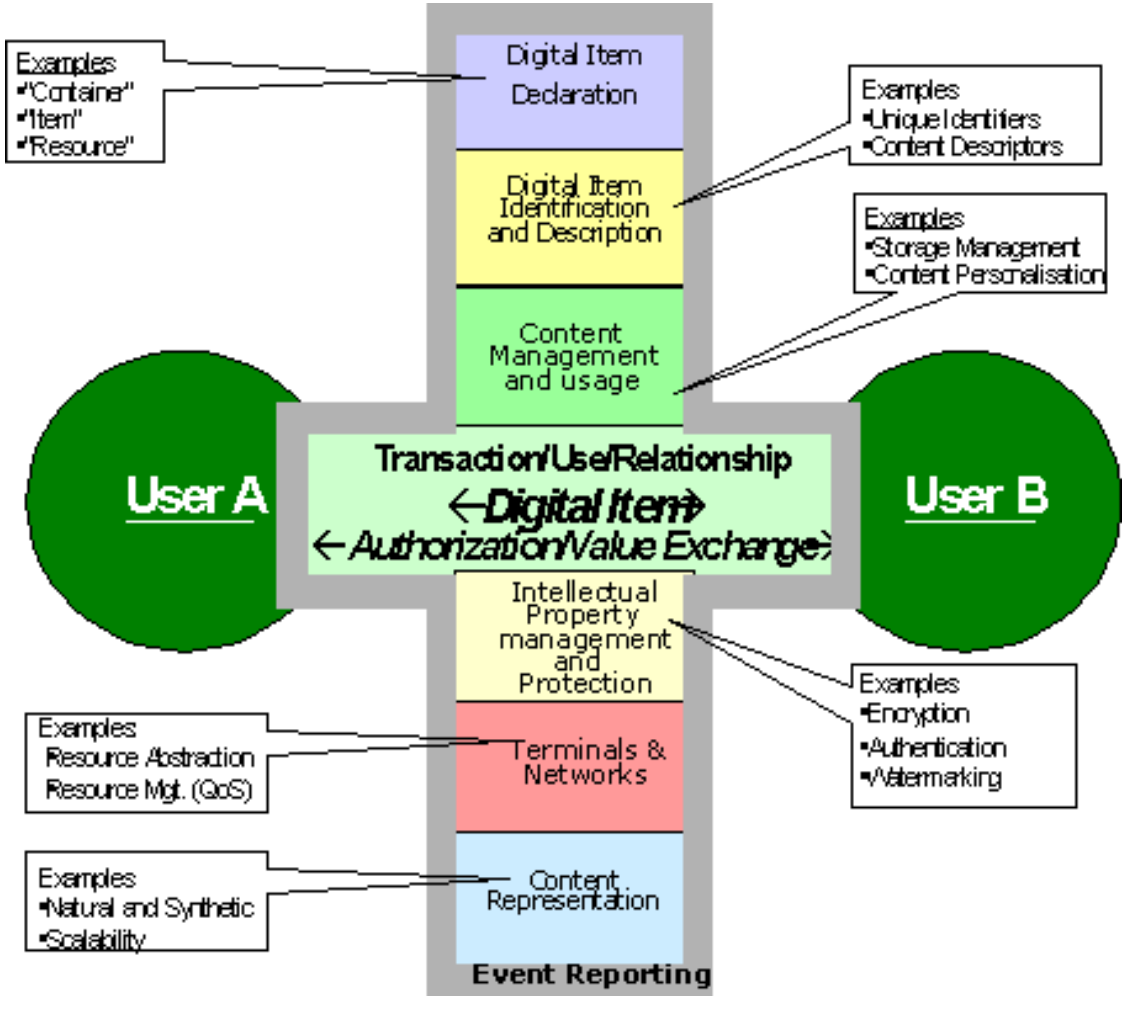
MPEG-7 Schemas Übersicht



MPEG-7 Schemas Erläuterung

- Basic Elements: Datentypen, Packages, Text, Zeit, Personen, Orte, . . .
- Content Management
  - Creation Information: Autoren, Drehort, etc.
  - Usage Information: Rechte, Preis, Alterslimit, . . .
  - Media Information: Datenformat von Master und Kopien
- Content Description
  - Structural aspects: Einteilung der Daten in räumliche/zeitliche Segmente
  - Conceptual aspects: Einteilung der zugehörigen Realität in Zeiten, Orte, etc.
  - Die beiden Aspekte sind über "links" verknüpft
- Navigation and Access
  - Summaries: Zusammenfassende Daten zum Auffinden der gewünschten MM-Daten (hierarchisch oder sequentiell organisiert)
  - Views: Verschiedene Ausprägungen der MM-Objekte (z.B. Beobachtungspunkt oder Bildqualität)
  - Variations: Sprachvarianten, Zusammenfassung eines Berichts, . . .
- Content Organization: Kollektionen von MM-Objekten
- User Interaction: Information über Präferenzen des Users, mittels derer nach MM-Objekten gesucht werden kann

MPEG-21 Übersicht



MPEG-21 Digital Item Declaration - Beispiel

```
<DIDL>
  <ITEM>
    <CHOICE MIN_SELECTIONS="1" MAX_SELECTIONS="1">
      <DESCRIPTOR>
        <STATEMENT TYPE="<some preamble>/text/text">
          What format would you prefer?
        </STATEMENT>
      </DESCRIPTOR>
      <SELECTION SELECT_ID="MP3_FORMAT">
        <DESCRIPTOR>
          <STATEMENT TYPE="<some preamble>/text/text">I want MP3</STATEMENT>
        </DESCRIPTOR>
      </SELECTION>
      <SELECTION SELECT_ID="WMA_FORMAT">
        <DESCRIPTOR>
          <STATEMENT TYPE="<some preamble>/text/text">I want WMA</STATEMENT>
        </DESCRIPTOR>
      </SELECTION>
    </CHOICE>
    <COMPONENT>
      <CONDITION REQUIRE="MP3_FORMAT"/>
      <RESOURCE REF="clip.mp3" TYPE="<some preamble>/audio/mp3"/>
    </COMPONENT>
    <COMPONENT>
      <CONDITION REQUIRE="WMA_FORMAT"/>
      <RESOURCE REF="clip.wma" TYPE="<some preamble>/audio/wma"/>
    </COMPONENT>
  </ITEM>
</DIDL>
```

MPEG-21 Hauptbestandteile

- Digital Item Declaration
  - Sprache, um alle möglichen Objekte anzuordnen
  - Strukturierung in Hierarchie (Container), Option (Choice), etc.
  - Angabe von Kommentaren, Hinweisen, Auswahltext, Verknüpfung zu Resourcen
  - basierend auf XML: DIDL = Digital Item Declaration Language
- Digital Item Identification and Description
  - Zuweisen von einheitlichen IDs, Namen, Beschreibung, etc.
  - Versionsmanagement
- Content Handling and Usage
  - Festlegung von User-Preferences
  - Auffinden von relevanten Daten im Netzwerk
  - Änderungshistorie
  - Intelligent Agents
- Intellectual Property Management and Protection: Angabe von Rechten, Verschlüsselung, Zugriffskontrolle
- Terminals and Networks: QoS (Quality of Service = Bildqualität, Delay, Jitter), APIs, NPIs
- Content Representation: Verwenden von MPEG-1,2,4,7 und anderen Formaten
- Event Reporting: Verfolgen von Fehlern und Performance