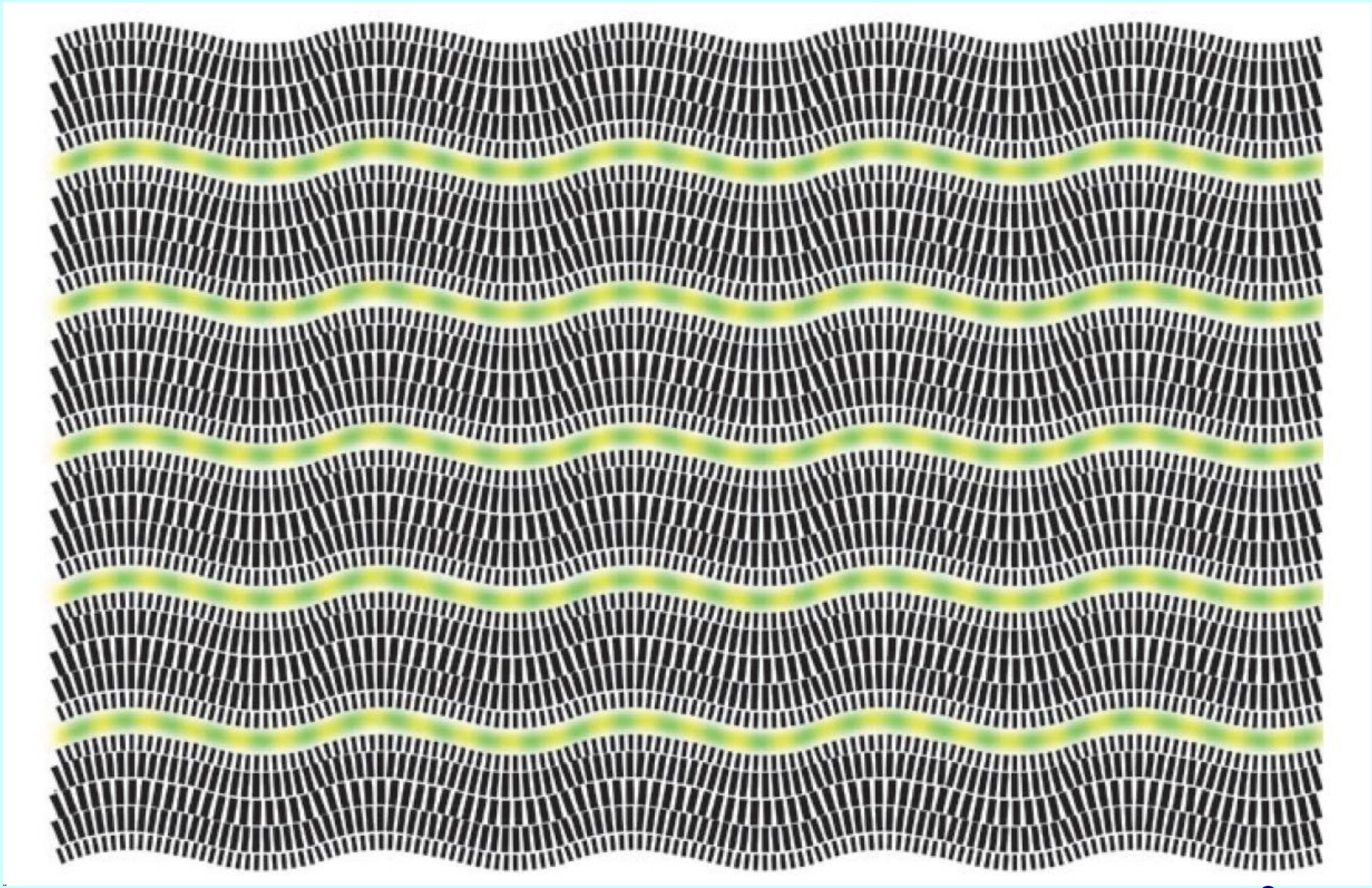
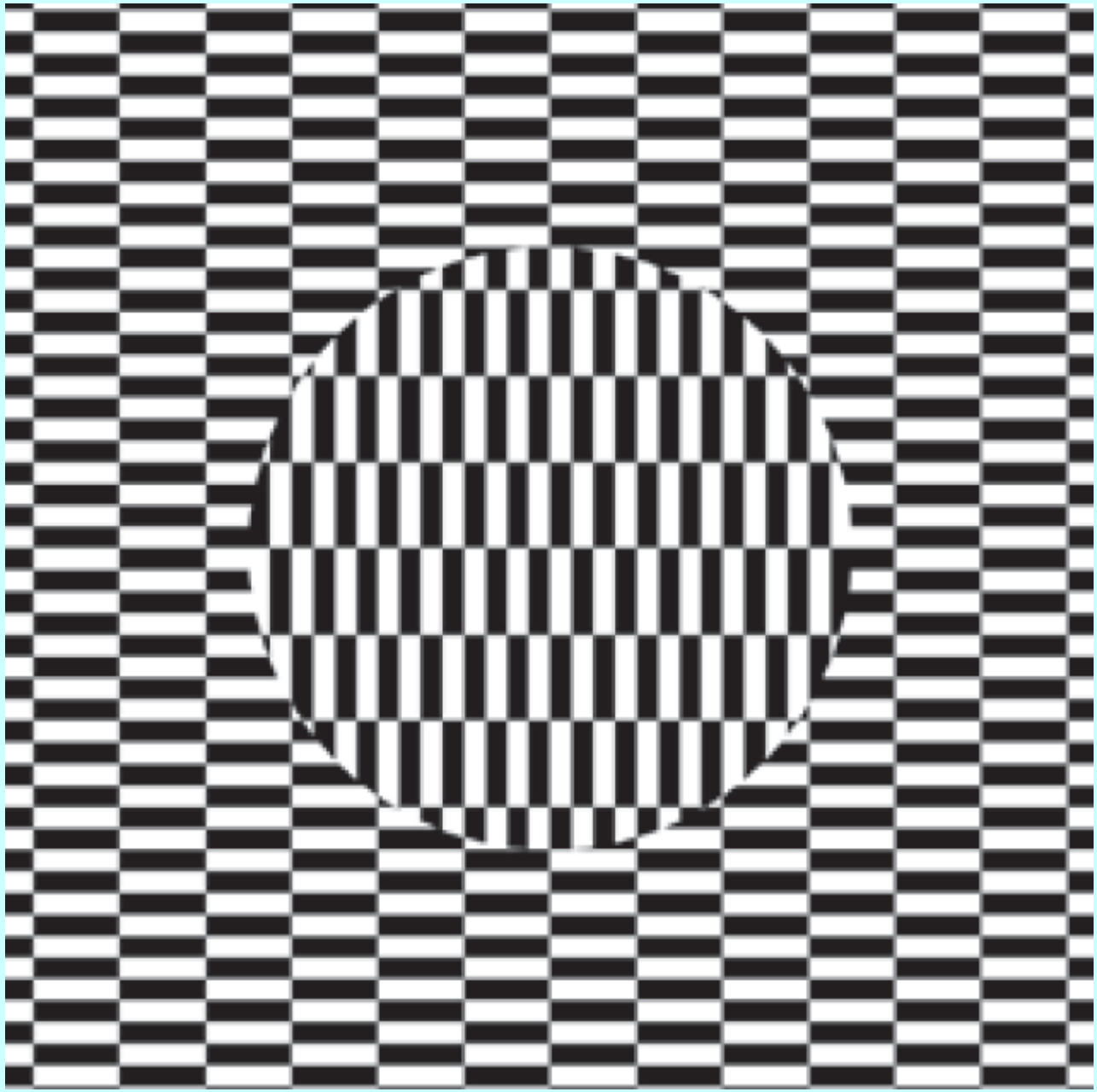
Module 10
Digital Video II

- Block Motion Estimation
- Interframe Coding and Motion Compensation
- Video Compression Standards
- MPEG-1
- MPEG-2
- MPEG-4/H.264

QUICK INDEX





Block Motion Estimation

- **Motion estimation** (flow) is important for **many** video applications: such as filtering, motion compensation, and compression.
- Practical systems use **block motion estimation**.
- Assumes the video motion is **moving blocks**.
- Blocks are assumed to **translate**.
- We can use **windows** and **windowed sets** to express this.

Video Notation

- A **video sequence I** is a 3-D array or signal:

$$\mathbf{I} = [I(i, j, k);\ 0 \le i \le N\text{-}1,\ 0 \le j \le M\text{-}1]$$

- In this Module, we regard **still images** as single images taken from video. Thus video consists of a sequence of still images:

$$\mathbf{I} = [\ \cdots \quad \mathbf{I}_{k-1} \quad \mathbf{I}_{k} \quad \mathbf{I}_{k+1} \quad \cdots\ ]$$

Video Windows

- A window **B** is a set of **2-D coordinate shifts** $\mathbf{B}_i = (m_i, n_i)$:

$$\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_{2M+1}\} = \{(m_1, n_1), ..., (m_{2M+1}, n_{2M+1})\}$$

- Given an image $\mathbf{I}_k$ and a window **B**, the **windowed set** at (i, j, k) is

$$\mathbf{B}\Diamond\mathbf{I}(i, j, k) = \mathbf{B}\Diamond\mathbf{I}_k(i, j) = \{I(i-m, j-n, k); (m, n) \in \mathbf{B}\}$$

  the **set of image pixels** covered by **B** at coordinate (i, j) at time k.

- The windows used are **SQUARE** and **nonoverlapping** for **ease of implementation**.

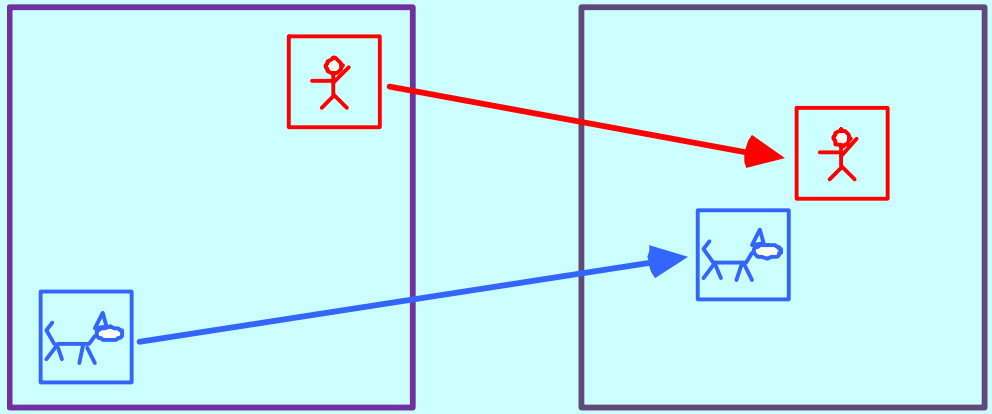
- In the standards, the blocks are commonly **16 x 16**.

Translational Block Motion

- This assumes that at **some later time** k+r, each block or windowed set at time k has translated in the i- and j-directions.

$$\mathbf{B}\Diamond\mathbf{I}(i, j, k) = \mathbf{B}\Diamond\mathbf{I}(i+d_1, j+d_2, k+r)$$

for integer displacements (d_1, d_2) and time shift r, as depicted:



Discussion of Block Motion

- **Advantages** of translational block models:
  - Only one motion vector needed per block
  - Ease of hardware implementation

- **Disadvantages** of translational block models:
  - Inaccurate for other motions types: zoom, rotation, bending
  - Leads to visual "blocking effects" at low bitrates

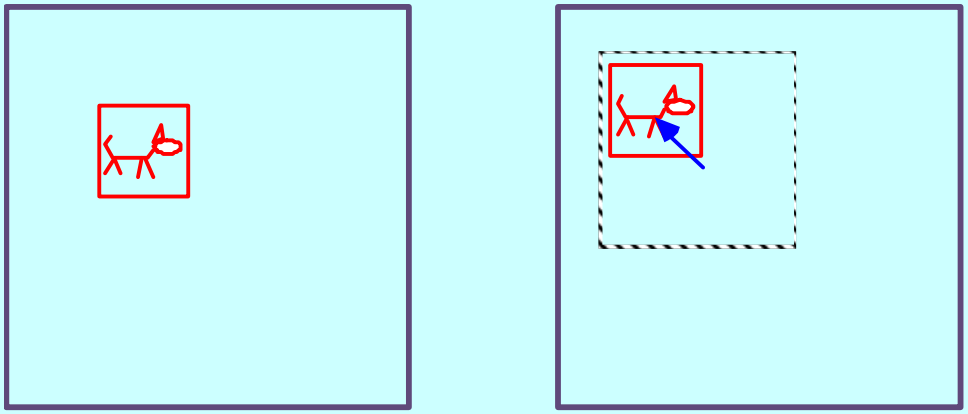
- It is possible to estimate other motion types, but at much higher cost. Standards (MPEG, H.261, etc.) use the simple model.

Block Matching

- **GOAL:** Estimate d_1, d_2 by **block matching** - the **simplest** method for estimating block motion.
- **Used by standards,** e.g. H.261 and MPEG-1, 2, 4.
- Involves a **simple search procedure** to find the best-fitting translational motion for each block.
- Method: **for each block B◊I**(i, j, k) in video signal **I** at time k, **search** for the **best-fitting block of the same size** at time k+1.
- The blocks that are found at time k+1 **may overlap..**

Search Space

- The search is conducted over a neighborhood centered around the location (i, j) of the original block:



frame k frame k+1

Block Match Measures

- **Goal**: Find block with the **minimum error** with respect to the original block:

$$\text{FIND: } \min_{(d_1, d_2)} \|\mathbf{B}\Diamond\mathbf{I}(i, j, k) - \mathbf{B}\Diamond\mathbf{I}(i+d_1, j+d_2, k+1)\|$$

where $\|\cdot\|$ is an error metric, such as (assume PxQ blocks):

$$\text{MSE}(d_1, d_2) = \frac{1}{PQ} \sum_{(m,n)\in \mathbf{B}} \sum [I(i-m, j-n, k) - I(i-m+d_1, j-n+d_2, k+1)]^2$$

$$\text{MAD}(d_1, d_2) = \frac{1}{PQ} \sum_{(m,n)\in \mathbf{B}} \sum |I(i-m, j-n, k) - I(i-m+d_1, j-n+d_2, k+1)|$$

- **MAD is commonly** used in practice - no computation of squares.

$$\text{Motion estimate: } (\hat{d}_1, \hat{d}_2) = \arg\min_{(d_1, d_2)} \text{MAD}(d_1, d_2)$$

Block Searching

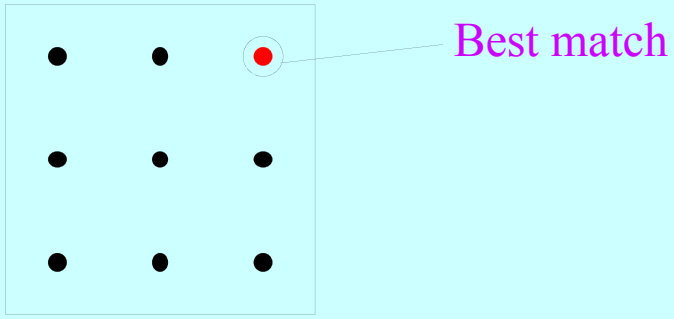
- In practice it is far too time consuming to check for **all** possible matches. Instead, a **subset** of possible matches is checked.
- First, the amount of translation is always limited:

$$-M \leq d_1, d_2 \leq M$$

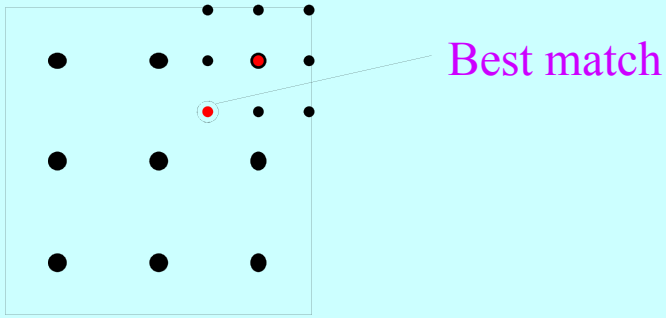
- **Three-step search** is a **typical** search strategy that is used. It involves narrowing down the best location using a directed search.
- However it is suboptimal.

Three-Step Search

- **Step 1:** Compute error at $d_1 = d_2 = 0$ at 9 **equally-spaced pixels**:

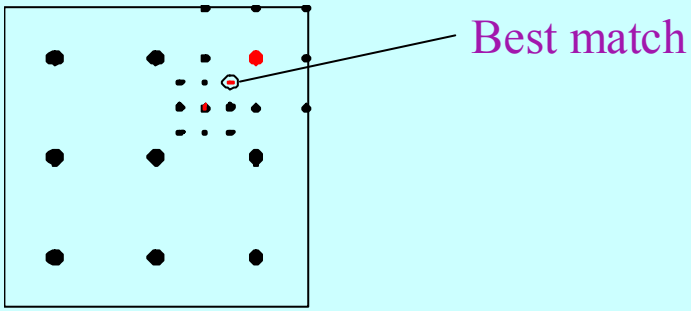


- **Step 2:** Localize the search near the best match from Step 1:

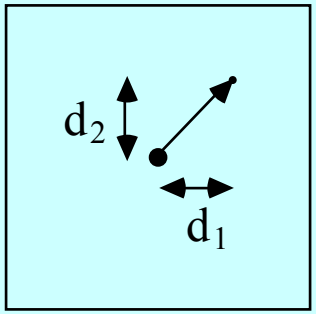


Three-Step Search (Cont'd)

- **Step 3:** Localize the search near the best match from Step 1:



- The **motion estimate** is then simply **the displacement** between the current block (time k) and the best match (time k+1):



Comments on Match Search

- We have looked at **forward** motion estimation: between current (k) and next (k+1) frames.
- **Backward** motion estimation can be used: between current (k) and last previous (k-1).
- Note that block-based motion estimation is really a form of **optical flow**.
- Current standards (e.g., H.261 and MPEG 1-2) use these simple methods. More sophisticated estimation methods are under study. These include
  - Rotational, scalable, and deformable block motions
  - Overlapping block motion estimation
  - Hierarchical motion estimation

Interframe Video Compression

- The data rates in digital video are **huge:**

  NTSC Color Digital Video ≈ 166 Mbps

  HDTV Color Digital Video ≈ 1.3 Gbps

  Super-35 (Theatre Quality) Video:
  (10 bits / color) x (3 colors/pixel) x (4096 pixels / line)
  x (3112 lines / frame) x 24 frames / second ≈ **9.18 Gbps**

- With **high-compression** video coding, these numbers become much more reasonable. Video contains a lot of **redundancy**.

Conditional Replenishment

- An old method underlying **the basic idea** of interframe coding. An image sequence is **differenced** along the time dimension.

- At each time k, find the between-frame difference:

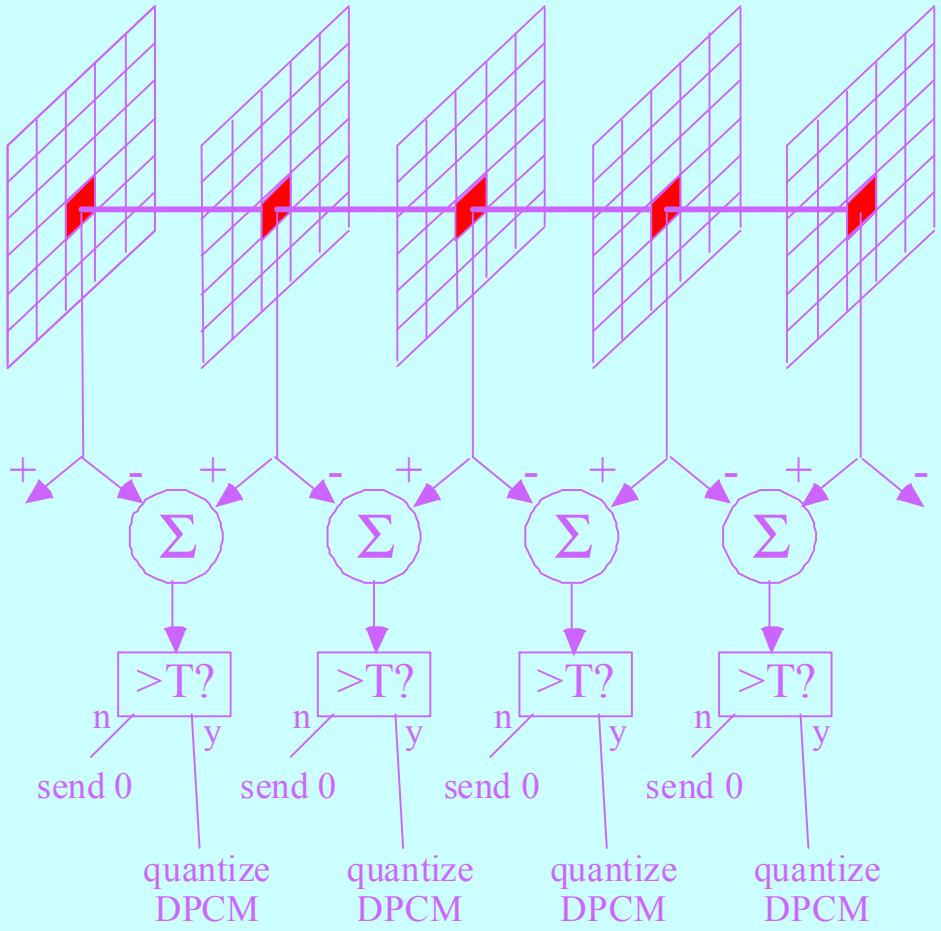
$$D(i, j, k) = I(i, j, k) - I(i, j, k-1)$$

- If above a threshold: $|D(i, j, k)| \geq T$ the difference is transmitted (**simple DPCM**). The differences may be quantized - **lossy DPCM**.

- If $|D(i, j, k)| < T$, then **zero difference** is transmitted (quantization). Since there may be many of these, they are **run-length coded**.

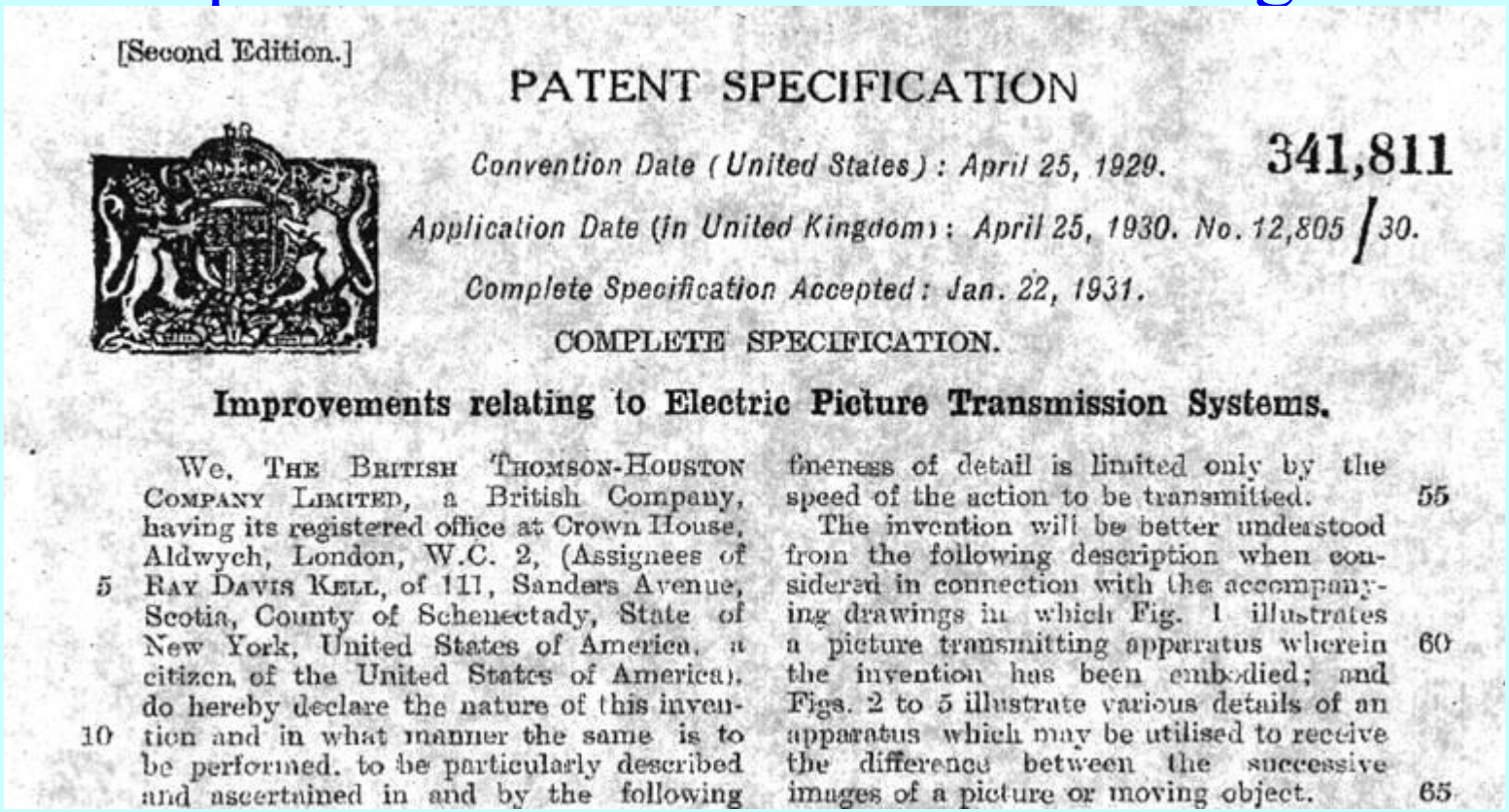
- Note that there is no use of **motion estimation**.

Conditional Replenishment



It's an old idea…

- In fact the first (1929) US Patent on video compression used frame differencing…



[Second Edition.]

PATENT SPECIFICATION

Convention Date (United States): April 25, 1929. **341,811**

Application Date (in United Kingdom): April 25, 1930. No. 12,805 / 30.

Complete Specification Accepted: Jan. 22, 1931.

COMPLETE SPECIFICATION.

Improvements relating to Electric Picture Transmission Systems.

We, THE BRITISH THOMSON-HOUSTON COMPANY LIMITED, a British Company, having its registered office at Crown House, Aldwych, London, W.C. 2, (Assignees of RAY DAVIS KELL, of 111, Sanders Avenue, Scotia, County of Schenectady, State of New York, United States of America, a citizen of the United States of America), do hereby declare the nature of this invention and in what manner the same is to be performed, to be particularly described and ascertained in and by the following

fineness of detail is limited only by the speed of the action to be transmitted.

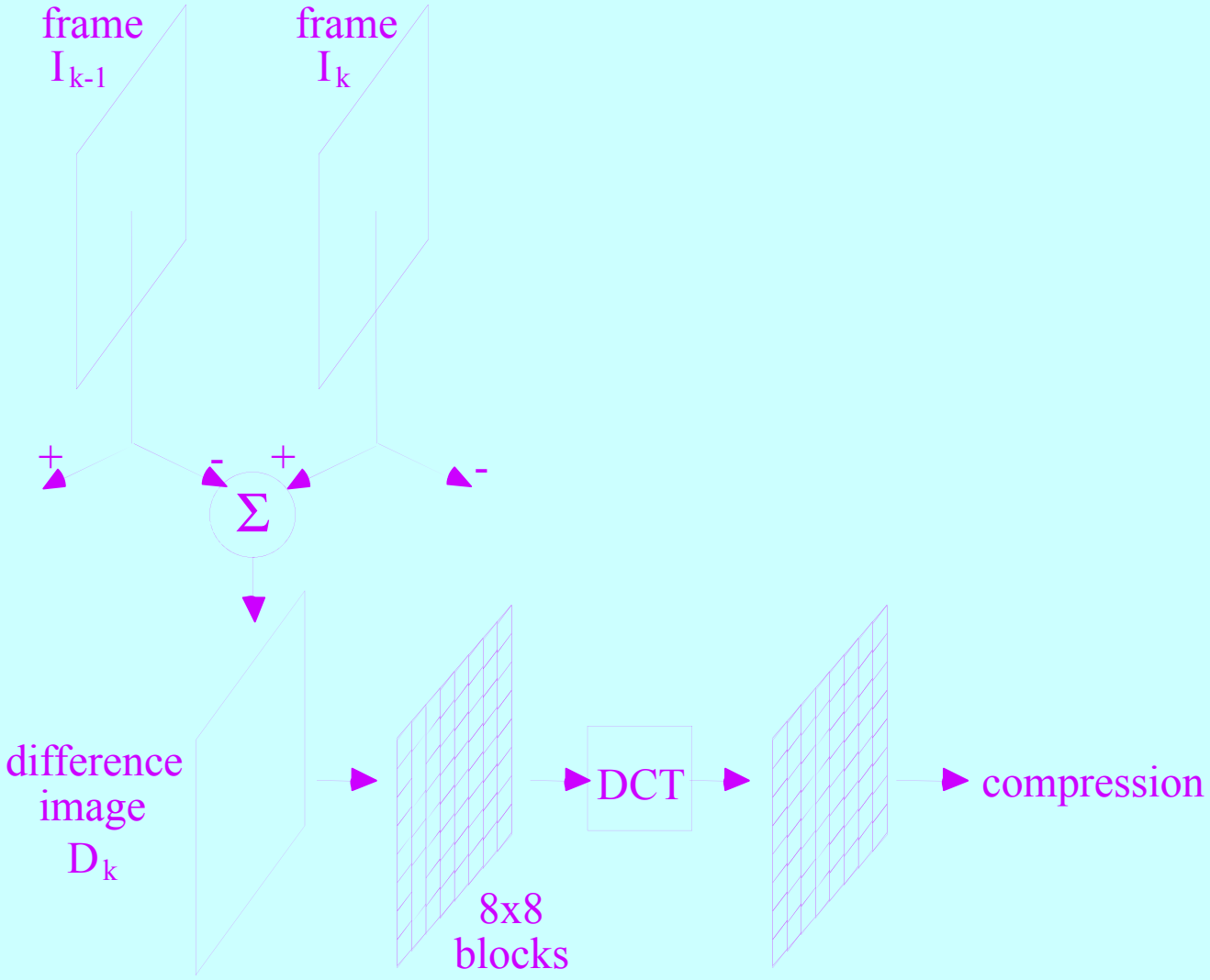
The invention will be better understood from the following description when considered in connection with the accompanying drawings in which Fig. 1 illustrates a picture transmitting apparatus wherein the invention has been embodied; and Figs. 2 to 5 illustrate various details of an apparatus which may be utilised to receive the difference between the successive images of a picture or moving object.

- See the whole thing here

Interframe Transform Coding

- A **big improvement** on DPCM is **block transform coding** (generally DCT-based) of image differences.
- Conceptually simple. The **difference images** $\mathbf{D}_k = [D(i, j, k)]$ are **broken into blocks**, which are then **DCT-coded** and **compressed**. Compression is generally **JPEG-like**.
- The **first image** is not differenced - just DCT-coded.

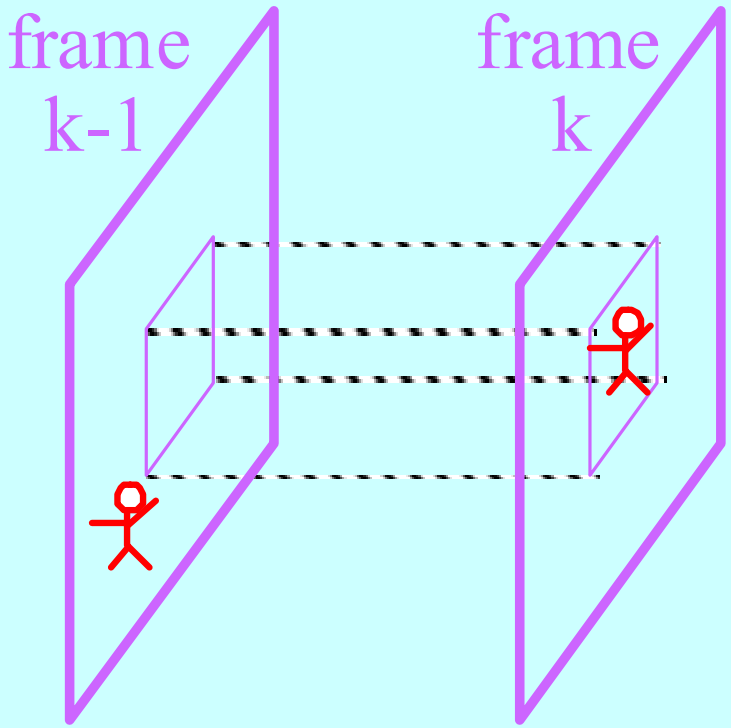
Interframe Transform Coding



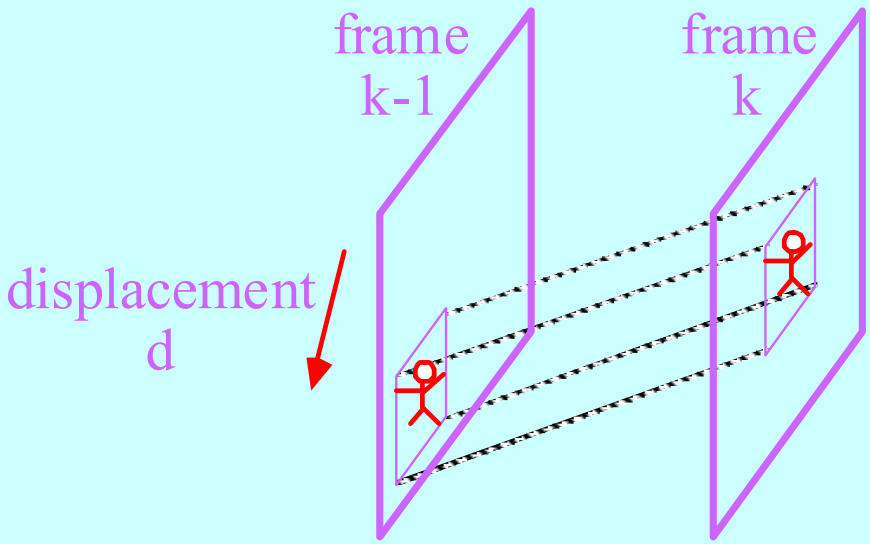
Motion Compensation

- **Problem**: Simple interframe DPCM and interframe transform coding do not effectively code **motion** very well.
- Although **changes** are detected and coded, the **high correlation** that exists between **intensities** that have **moved between frames** is not exploited. Thus, **lower compression** efficiency.
- The **computed differences** in the preceding schemes will often be taken between subimages of **different** physical objects/regions.
- This is where **motion estimation methods** become valuable. Motion estimation yields **displacement vectors** between frames.

Motion Displacement



Motion-Compensated Displacement



computed displacement identifies
corresponding blocks

- **Motion compensation** uses these displacements to compute differences between corresponding blocks - along the **motion trajectory.**

Motion-Compensated Transform Coding

- Involves the following generic steps:

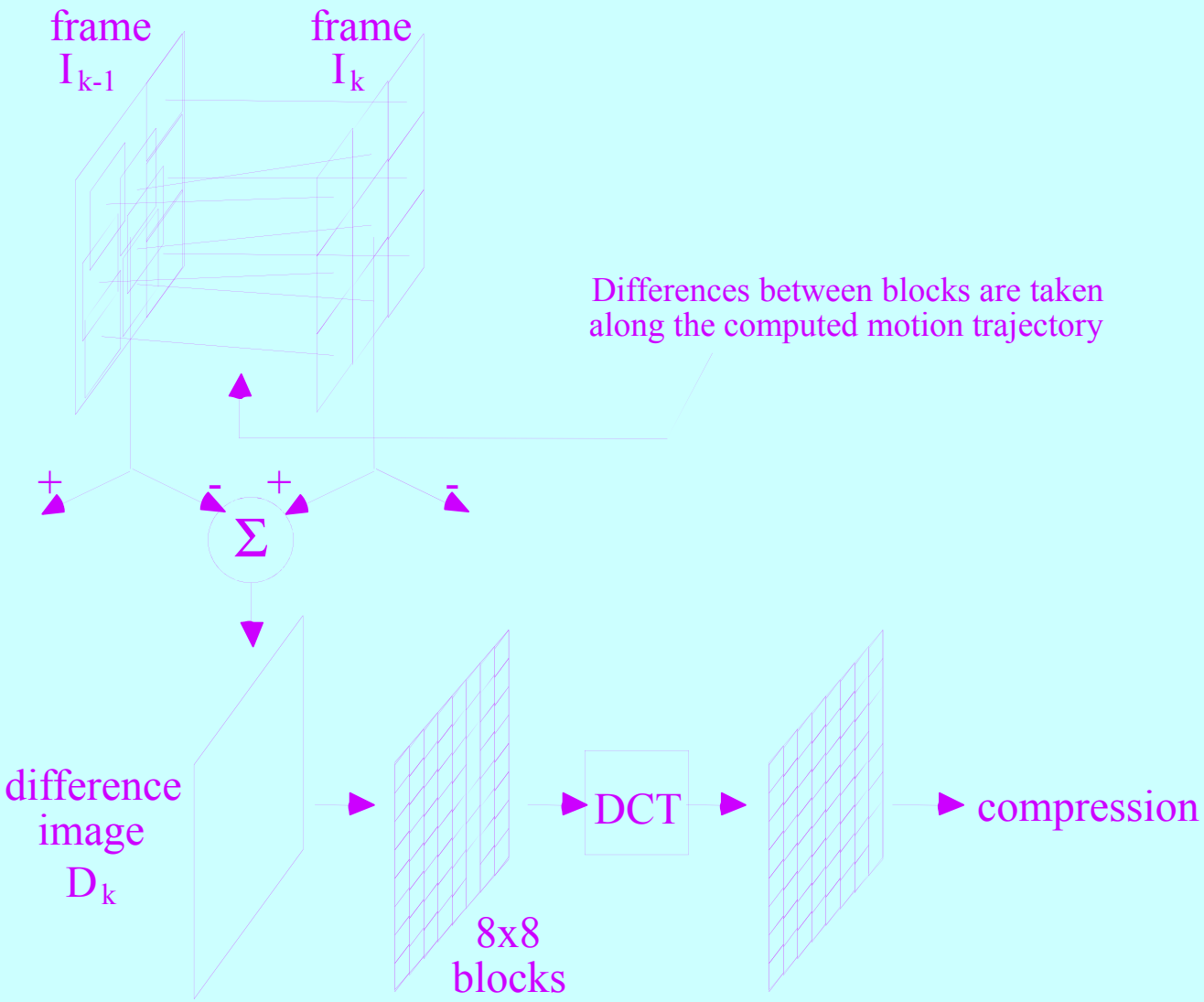
  (1) Compute block motion displacement vectors using **lookback** from frame $\mathbf{I}_k$ to frame $\mathbf{I}_{k-1}$. Usually 16x16 blocks. Note that the blocks are non-overlapping in frame $\mathbf{I}_k$.

  (2) Compute **motion-compensated difference image** $\mathbf{D}_k$ by differencing each 16x16 block in $\mathbf{I}_k$ with it's corresponding displaced block in $\mathbf{I}_{k-1}$.

  (3) Subdivide difference image $\mathbf{D}_k$ into sub-blocks (usually 8x8) and code using **JPEG-like algorithm**.

  (4) The **first image** is simply JPEG-coded for reference.

Motion-Compensated Transform Coding



Comments on Motion Compensation

- **Motion compensation (MC)** is highly effective for

  - **Increasing compression efficiency** - More redundancy is exploited between matched motion blocks

  - **Reducing "ghosting" artifacts** in compressed video - Very fast movements result in large differences between blocks. Without MC, compression of the large differences can result in "**motion ghosts.**"

  - Accommodating compression to **temporal aliasing** - something that the human eye does very well (and which it is quite sensitive to if not used in compression!)

- Video compression standards such as H.261, MPEG-1 and MPEG-2 **all use motion-compensated transform coding** based on block motion estimation and the DCT.

Three-Dimensional Transform Coding

- A **digital video** is a 3-D digital signal I(i, j, k) that can be broken into "data cubes" (3-D blocks).

- Assuming 8x8x8 data cubes, compute 8x8x8 3-D DCTs.

- One can imagine **JPEG-like compression** for video based on this.

Disadvantages:

- Requires storage of 8 frames, hence a **delay**
- Doesn't exploit motion redundancy as well as MC methods
- Can suffer from ghosting
- **Not currently used!**

Three-Dimensional Wavelet Coding

- **Wavelet transform** methods can also be extended to 3-D.
- Such a decomposition should use **few bands** (perhaps no more than **two**) along the **time dimension** to reduce **frame storage**.
- An **active research area**. How to embed **motion compensation** into the wavelet code is a **hot topic**.

Advantages:

- **No blocking** effects

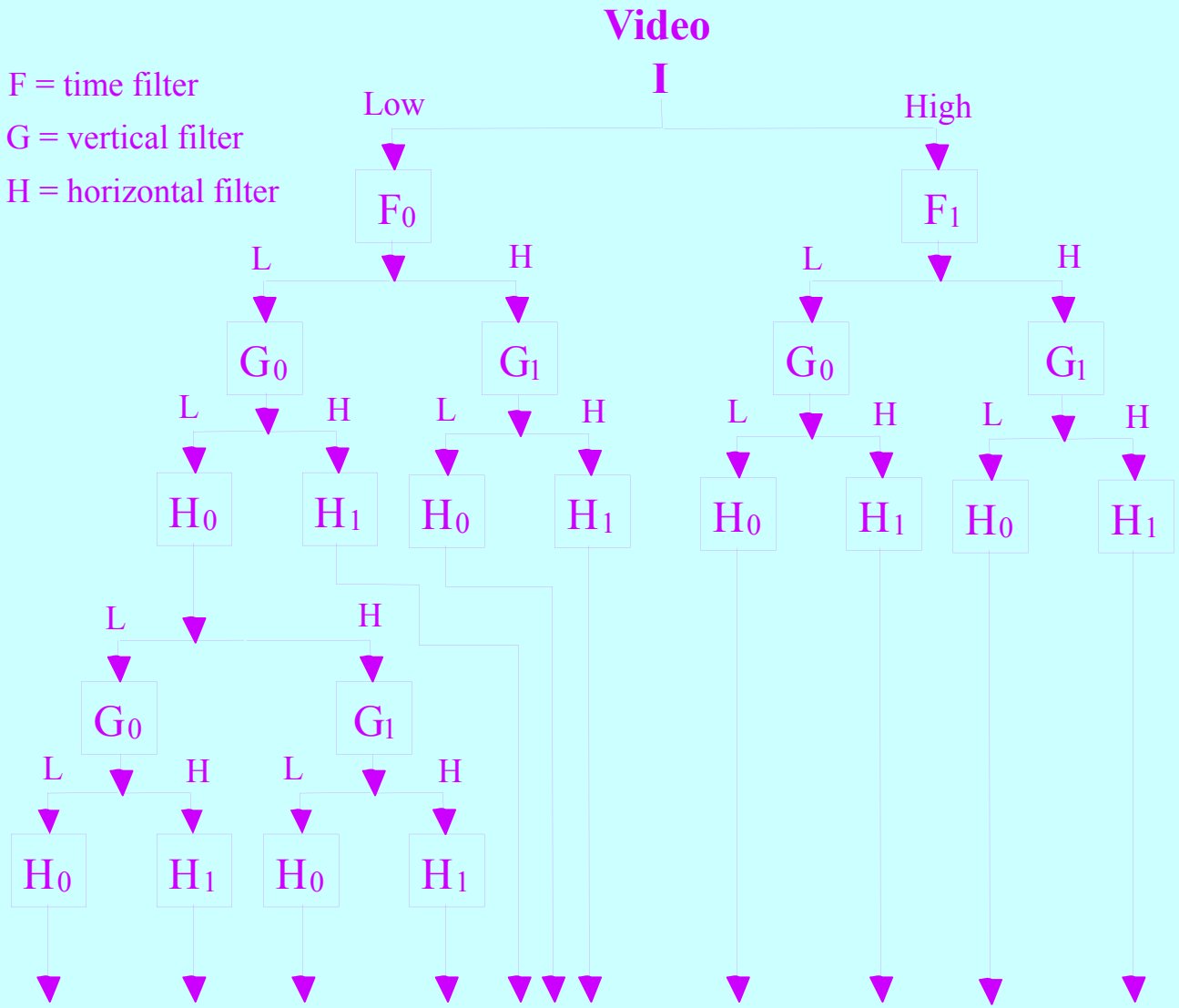
- **Scalable** - meaning that the coded video is easily available at a variety of resolutions

Disadvantages:

- Not clear how to incorporate motion compensation

- **Very Promising** in the long run!

Example 11-band DWT



Video Compression Standards

- We review several important video compression standards. All of these standards are notable in that they **do not describe how the video codec** (coder-decoder) **functions** – but rather, they exactly **specify the syntax** of the compressed videos.

MPEG-1

- Standardized in early 1990's by ISO **M**oving **P**icture **E**xpert **G**roup
- Intended for storage/retrieval of digital video at ≈ **1.5 Mbps**
- Target applications: **Multimedia:** Video CD and hard disk storage

MPEG-2

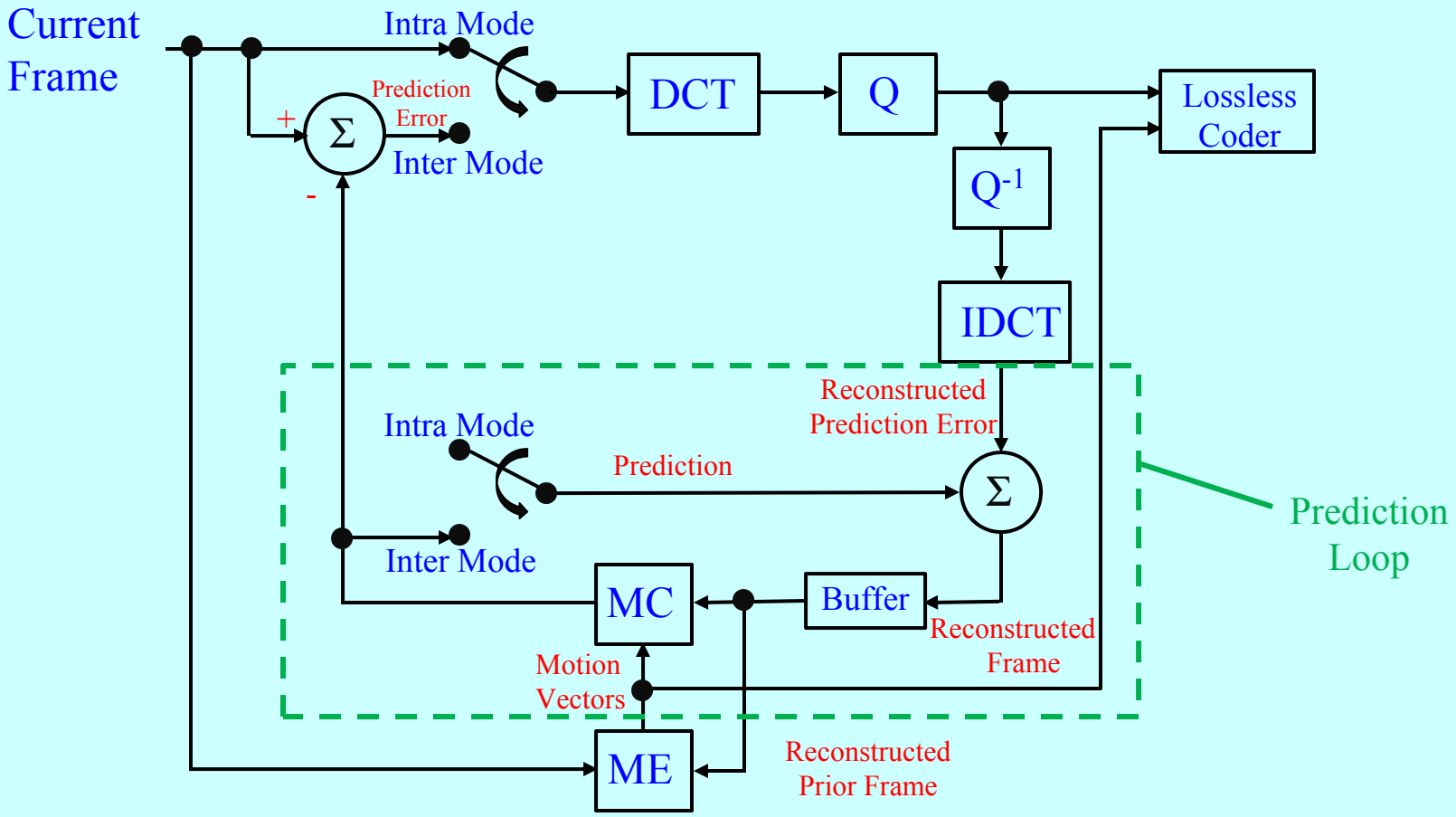
- Standardized in mid-1990's by MPEG Committee
- Intended for storage/retrieval/transmission of video at >3 Mbps, but more typically 10-80 Mbps
- Target applications: Digital cable, satellite TV, DVD, high-definition TV (HDTV)

Video Compression Standards

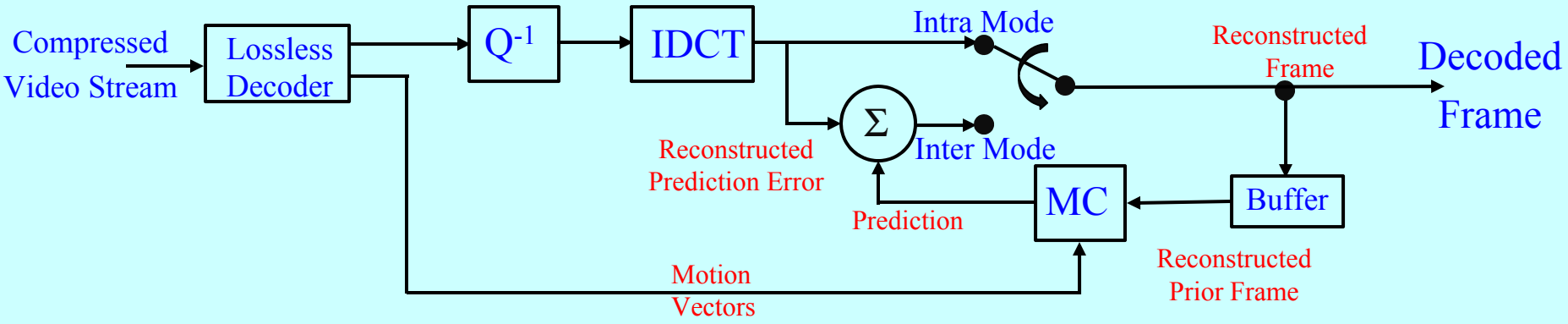
MPEG-4, Part 10 (AVC) / H.264

- **Standardized** in 2003.
- A large, **complex** video/audio standard.
- We will review only **Part 10**, which offers **high-quality video coding**.
- **MPEG-4**, **Part** 10 (AVC) is identical to the **H.264** video compression standard.
- Our survey will be very high-level!!

Generic Encoder Diagram



Generic Decoder Diagram



Interframe Prediction Principles

- **Decoder predicts current frame** using **residual signal** and **motion vectors**. It is the sum of the current residual and the MC prediction of the previous frame.
- The **residual signal** is the difference between **current frame** and **motion compensated prediction** of the **previous frame**.
- This special design ensures that the **predictor** in the **encoder** and in the **decoder** are **identical** and receive the **same inputs.**
- If **no quantization,** then this is a **lossless coder-decoder combination.**
- **With quantization**, this prevents the predictors from **straying** from each other (**error propagation** over time).

MPEG-1

- Easy **access to individual frames** from the compressed video. This is done by defining "independent access points," or **I-frames**.
- **Reverse playback** and **Fast Forward / search** capability
- Coding/decoding delay of up to 1 second - deemed acceptable for multimedia-type applications.

MPEG-1 Breakdown

- MPEG-1 consists of several parts:
  1. Synchronization and multiplexing of video and audio
  2. **Compression codec for non-interlaced video signals**
  3. Compression codec for perceptual coding of audio signals. The standard defines three "layers," or levels of complexity, of MPEG audio coding.

     MP1 or MPEG-1 Part 3 Layer 1

     MP2 or MPEG-1 Part 3 Layer 2

     MP3* or MPEG-1 Part 3 Layer 3
  4. Procedures for testing conformance.
  5. Reference software

*e.g., ipod

MPEG-1 Compression

- Two compression modes: **intraframe** and **interframe**.
- **Intraframe** (within frame) is basically JPEG-like
- **Interframe** uses DCT coding of difference blocks, **with or without** motion compensation (MC is optional).

MPEG-1 Picture Hierarchy

- In MPEG-1, the video data has a **hierarchical** structure:
  - **Blocks** are 8x8 pixel arrays. They are used by the DCT.
  - **Macroblocks** are 16x16 pixel arrays composed of 4 **blocks**. Actually consists of 16x16 **luminance** pixels and 8x8 Cr and 8x8 Cb **chrominance** pixels.
  - **Pictures** are composed of **macroblocks**. The pictures are compressed in different ways according to type:
    - **D-pictures** contain only the DC component of each block, hence yield very low resolution (low bitrate) browsing.
    - **I-pictures** are **intra-frame coded** via JPEG-like DCT.
    - **P-pictures** and **B-pictures** are **inter-frame** MC DCT-based time-difference coded. However, they differ:
      - **P-pictures** use only lookback motion estimation, always with respect to a preceding **I-** or **P-picture**.
      - **B-pictures** use both lookback, lookahead, and bi-directional motion estimation, always with respect to **I-** or **P-pictures** (even if several frames ahead/behind)
      - **Video sequences** are composed of **pictures**.

MPEG-1 Intraframe Compression

- **I-pictures** are always intraframe coded very similar to JPEG.
- **DCT** of 8x8 **blocks,** followed by **quantization** using a normalization array, creating **numerous zero coefficients**.
- The **default** MPEG normalization matrix:

$$\left(\frac{1}{\text{MQUANT}}\right)\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

- May be **user-specified** (& transmitted). Note similarity to (except for a factor of 2) and differences relative to JPEG array in Module 7.

MPEG-1 Intraframe Compression

- The factor MQUANT is different from JPEG - it may be varied from block-to-block for higher compression (larger MQUANT if the block is “busy”). This allows **adaptive quantization**.
- Coefficients are **zig-zag ordered** and zeros are **run-length coded**.
- DC coefficients from each **block** are DPCM-coded as in JPEG.

MPEG-1 Interframe Compression

- A **motion estimate** is computed over each **16x16 luminance macroblock**, and the **8x8 MC block differences** (luma &chroma) are DCT-coded and compressed.

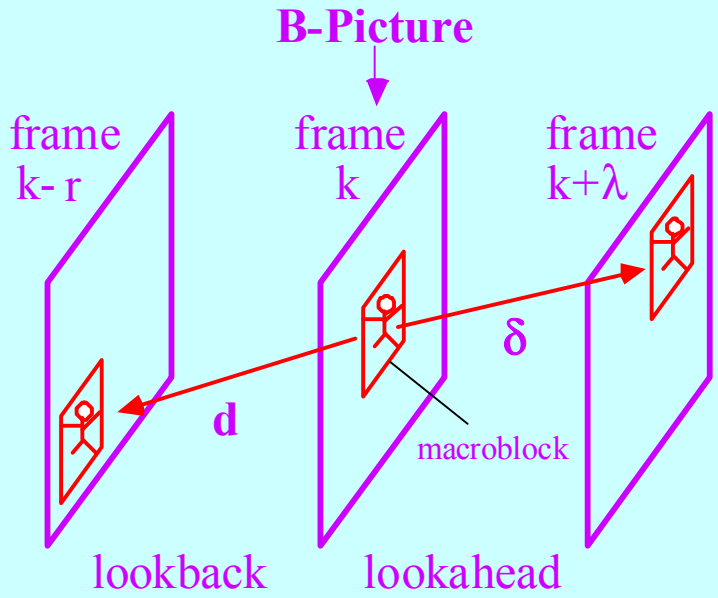
- **I-pictures** may **not** be inter-frame coded.

- **Macroblocks** in **P-pictures** are examined to decide whether to:
  - **Intraframe code** (if insufficient displacement)
  - **Interframe code** wrt previous frame as above
  - **Skip** (if insufficient displacement **and** change)

  No recommendation given how to select the coding mode.

- **Macroblocks** in **B-pictures** are examined to decide whether to intraframe code, interframe code, or skip frame. **Both lookback** and **lookahead** (possibly more than 1 frame) motion estimation can be accomplished for B-pictures.

Look-back &Look-ahead Modes



B-pictures in MPEG-1

- Consider macroblock $\mathbf{B}\Diamond\mathbf{I}(i, j, k)$. The displaced regions in frames k-r and k+l are:

  $\mathbf{B}\Diamond\mathbf{I}(i-d_1, j-d_2, k-r)$ and $\mathbf{B}\Diamond\mathbf{I}(i-\delta_1, j-\delta_2, k+l)$

- The following frame differences may be selected from and subsequently compressed:

  Lookback: $\mathbf{B}\Diamond\mathbf{D}(i, j, k) = \mathbf{B}\Diamond\mathbf{I}(i, j, k) - \mathbf{B}\Diamond\mathbf{I}(i-d_1, j-d_2, k-r)$

  Lookahead: $\mathbf{B}\Diamond\mathbf{D}(i, j, k) = \mathbf{B}\Diamond\mathbf{I}(i, j, k) - \mathbf{B}\Diamond\mathbf{I}(i-\delta_1, j-\delta_2, k+l)$

  Bidirectional: $\mathbf{B}\Diamond\mathbf{D}(i, j, k) = \mathbf{B}\Diamond\mathbf{I}(i, j, k) - [\mathbf{B}\Diamond\mathbf{I}(i-d_1, j-d_2, k-r) + \mathbf{B}\Diamond\mathbf{I}(i-\delta_1, j-\delta_2, k+l)]/2$

- **Which** to use is **not recommended** in MPEG-1. However, for example, the one with the **smallest variance** might be used (common).

- If **interframe** coding is selected, then 2 displacements vectors **d** and **δ** must be transmitted along with $\mathbf{B}\Diamond\mathbf{D}(i, j, k)$.

B-pictures in MPEG-1

- Hence macroblocks in **B-pictures** may be:
  - **intraframe coded** (if insufficient displacement)
  - **lookback interframe coded** wrt previous frame
  - **lookahead interframe coded** wrt next frame
  - **bidirectional interframe coded** wrt next/prev frames
  - **skip** (if insufficient displacement **and** change)

B-Pictures

- Main advantages of **B-pictures**:
  - **Occlusion**: Object occlusion problems may be avoided by selecting lookback vs. lookahead estimation. If an object is partially occluded in lookback picture, then use lookahead.
  - **Noise reduction**: Averaging of lookback and lookahead may **reduce noise** - significant when **differencing**.

- Main disadvantages of **B-pictures**:
  - **B-pictures** require two frames of previous/future pictures.
  - If many **B-pictures** are used, then (since motion estimation for both **P-pictures** and **B-pictures** must be with respect to **I-** or **P-Pictures**), there will be **reduced temporal correlation**.

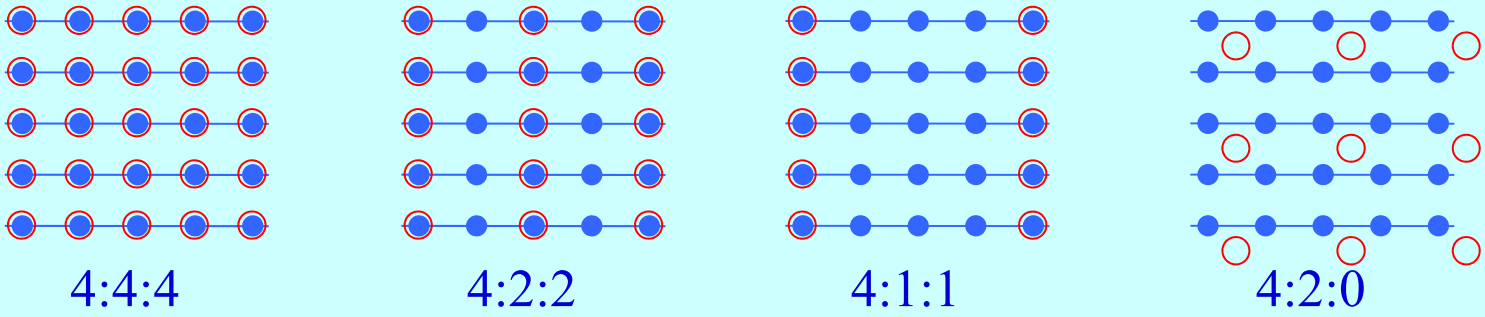
Chroma Sub-Sampling in MPEG

- MPEG uses the **Y-Cr-Cb** color representation.
- Since color is an **area attribute**, it can be sampled more heavily than luminance – a very old strategy.
- Sampling formats:
  - 4:4:4 – sampling rates of Y, Cr, Cb are the same
  - 4:2:2 – sampling rates of Cr, Cb are ½ that of Y
  - 4:1:1 – sampling rates of Cr, Cb are ¼ that of Y
  - 4:2:0 – sampling rates of Cr, Cb are also ¼ that of Y

Y-Cb-Cr Sampling

○ = Cr, Cb samples

● = Y samples



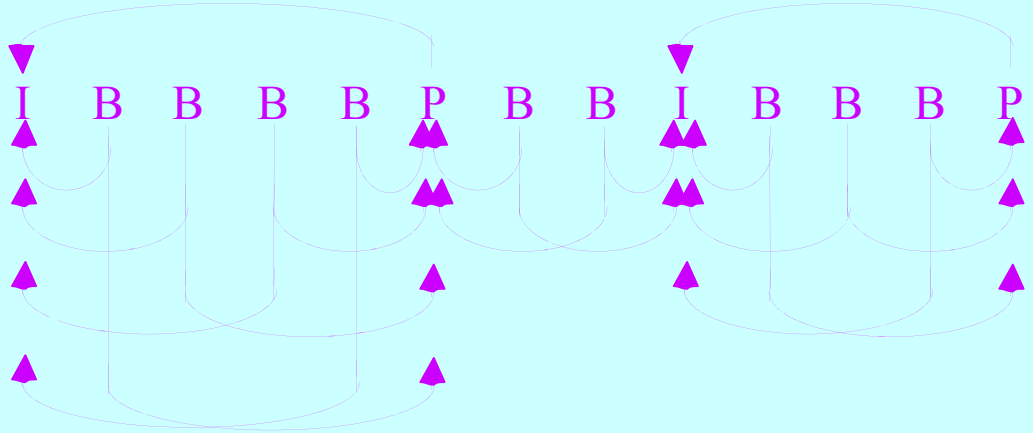
- Note that the chroma 4:2:0 samples are **calculated** for storage or transport. For **display**, 4:2:2, 4:1:1, and 4:2:0 samples are **interpolated** back to 4:4:4.

Order of Coding in MPEG-1

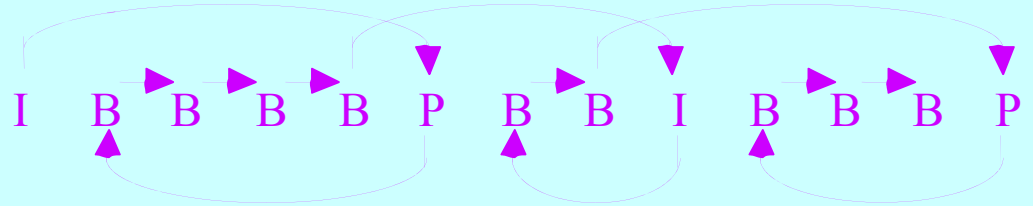
- **B-pictures** are optional (else not much different from H.261).
- Video always starts with an **I-picture**. One of every 132 frames must be an **I-picture**. This reduces accumulated errors.
- If **B-pictures** are used, **pictures** are not coded in time-order because of lookback/lookahead options. Motion estimation is always computed with respect to **I-** or **P-pictures** even if several frames away.
- The lookahead and lookback relationships between **I-**, **P-**, and **B-pictures** in a sequence of 13 pictures are shown:

Example MPEG-1 Coding Order

- The lookahead and lookback relationships between **I-**, **P-**, and **B-pictures** in a sequence of 13 pictures are shown:



- The following diagram depicts the **coding order** of the same sequence.



Sample MPEG-1 Videos

- Of course, everyone is pretty familiar with these nowadays (e.g. YouTube)

Hale-Bopp

Beardo-Wierdo

Kitty Cat

Shoemaker-Levy

Orion Nebula

Mars

MPEG-2

- MPEG-1 does **not** supply sufficiently good **quality** for most **entertainment** applications.
- MPEG-2 coding is **substantially the same as** MPEG-1 but with increased capability.

MPEG-2 features:

- **Higher quality** (and high bitrates)
- **Interlaced video** input allowed
- **Scalability**

- Currently used in digital cable TV, satellite TV, DVD videos and HDTV.

MPEG-2 Suite

- Actually a 10-part standard:
  - Part 1 Systems - describes synchronization and multiplexing of video and audio
  - **Part 2 Video - compression codec for interlaced and non-interlaced video signals.**
  - Part 3 Audio - compression codec for perceptual coding of audio signals. A multichannel-enabled extension of MPEG-1 audio.
  - Part 4 Describes procedures for testing compliance.
  - Part 5 Describes systems for Software simulation.
  - Part 6 Describes extensions for DSM-CC (Digital Storage Media Command and Control.)
  - Part 7 Advanced Audio Coding (AAC)
  - Part 8: 10-bit video extension for studeo video Withdrawn due to lack of industry interest.
  - Part 9 Extension for real time interfaces.
  - Part 10 Conformance extensions for DSM-CC.

MPEG-2 Specs and Definitions

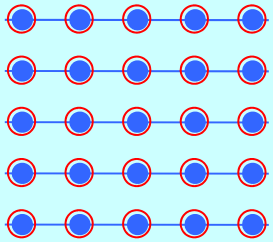
- **Macroblocks** consists of 16x16 **luminance** pixels and either: 8x8, 16x8 or 16x16 (Cr and Cb) **chrominance** pixels.

- **Frame pictures** are composed of both the even and odd components of interlaced video - **interleaved**. **Frame pictures** can be **I-**, **P-**, or **B-pictures**. If the input video is **progressive**, all pictures are of this type.

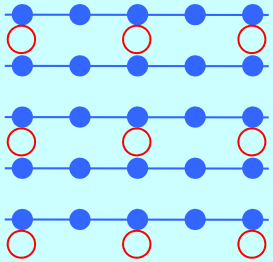
- **Field pictures** are even/odd fields regarded as separate pictures. Can be **I-**, **P-**, or **B-pictures**. Always appear in pairs called **top** and **bottom fields**, always of the same type (**I-**, **P-**, or **B-**).

- **Macroblocks** are defined for both **frame** and **field pictures**.

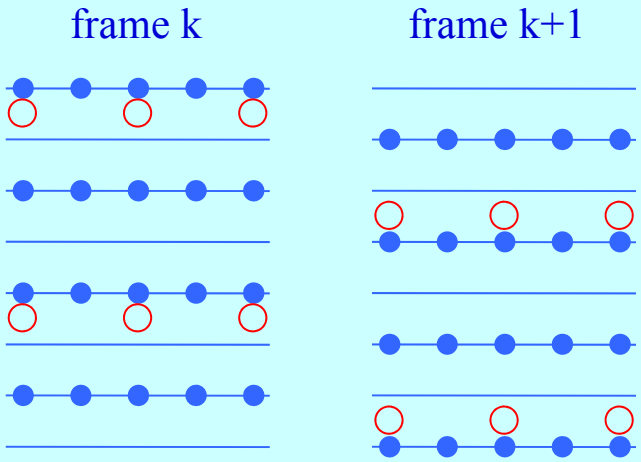
MPEG-2 Chroma Sampling



4:4:4



4:2:0
progressive



4:2:0
interlaced

MPEG-2

- Compression of video via MPEG-2 is **essentially identical** to MPEG-1.
- Except for handling of **interlacing**, which complicates every aspect.
- By the way, interlacing gets less important every day!
- And **scalability**, which we won't get into …. MPEG-2 still exists, but is on the way out.

MPEG-2 Videos

Dancers

Bicycle

Mobile

Scene Change

Ping Pong Ghosting

- Of course, these videos will have their appearance modified by the choice of codec, my laptop speed, and the display/projector

MPEG-4 / H.264

- MPEG-4/H.264 originally targeted for very low bitrate video transmission applications: **4.8 - 32 kbps**.
- **The leading codec choice for wireless video applications.**
- And increasingly for **cable/satellite.** Also **Blu-Ray.**

- **Excellent quality/compression performance** allows use over wide range of bitrates too (up to 960 Mbps), perhaps **digital television and digital cinema**.
- Incorporates **basic frame-based** MPEG-1,2 features.
- Same chroma sampling scheme as MPEG-2.

Overall MPEG-4 Breakdown

- MPEG-4 consists of **many** "parts":

1. Systems: Describes synchronization and multiplexing of video and audio.
2. **Visual: A compression codec for visual data. Similar to MPEG-1,2 but with object coding.**
3. Audio: A set of compression codecs for perceptual coding of audio signals.
4. Conformance: Describes procedures for testing conformance to the standard.
5. Reference Software: Software for demonstrating the standard.
6. Delivery Multimedia Integration Framework (DMIF).
7. Optimized Reference Software. Examples of improved implementations (see Part 5).
8. Carriage on IP networks: Specifies a method to carry MPEG-4 content on IP networks.
9. Reference Hardware: Hardware designs to implement the standard.
10. **Advanced Video Coding (AVC): A codec for video signals. Identical to ITU-T H.264.**
11. Scene description and Application engine; used for 2D & 3D interactive content.
12. ISO Base Media File Format: A file format for storing media content.
13. Intellectual Property Management and Protection (IPMP) Extensions.
14. MPEG-4 File Format: Designated container file format for MPEG-4 content.
15. AVC File Format: For storage of Part 10 video based on Part 12.
16. Animation Framework eXtension (AFX).
17. Timed Text subtitle format.
18. Font Compression and Streaming (for OpenType fonts).
19. Synthesized Texture Stream.
20. Lightweight Scene Representation (LASeR).
21. Graphical Framework eXtension (GFX).
22. Open Font Format Specification (OFFS) based on OpenType
23. Symbolic Music Representation (SMR).

MPEG-4 Part 2

- Supports **access, manipulation**, and **separate coding** of **objects**. First standard allowing **content-based video coding**. Instead of the prior frame-based compression, MPEG-4 Part 2 can divide the video into **Video Object Planes**.

- **Object access & manipulation modes** allow handling of multimedia / graphical videos. Modes allows for **mesh modeling and coding of graphical objects**; modeling and compression of **textured objects**; separate coding of unchanging objects ("**sprites**"); **interactivity** with the video stream.

- **Separate object coding** can give much higher compression - moving objects and motionless backgrounds can be compressed separately.

  Example: A moving car driving through a quiet city scene.

  Example Video Candidate for Object Plane Coding

- Yet, object-based coding introduces a big challenge: **segmenting** objects from background. This is very difficult. Most built codecs **don't use this stuff**.

Some of the Plethora of H.264* Features

Other aspects of MPEG-4 AVC are promoting its success.

- Highly flexible, **multi-picture inter-picture prediction**: Can use previously-encoded pictures as references - up to 32 frames - unlike B-pictures (2). In certain scenes - with rapid repetitive flashing or back-and-forth scene - very significant reductions in bit rate.
- **Variable block-size motion compensation** (VBSMC) with block sizes as large as 16×16 and as small as 4×4, enabling precise segmentation of moving regions.
- **Quarter-pixel precision for motion compensation**, enabling precise displacement vectors.
- **New transform design**: (1) An exact-match integer 4×4 spatial block transform (similar to DCT design, but simplified and with exactly-specified decoding. (2) An exact-match integer 8×8 spatial block transform, allowing highly correlated regions to be compressed more efficiently than with the 4×4 transform. (3) Adaptive encoder selection between the 4×4 and 8×8 transform block sizes for the integer transform operation (not supported in all profiles).
- An in-loop **de-blocking filter** the reduces blocking artifacts.
- **New VLC entropy coding designs**: (1) Context-adaptive binary arithmetic coding (CABAC) - losslessly compress syntax elements in the video stream knowing the probabilities of syntax elements in a given context; (2) Context-adaptive variable-length coding (CAVLC) - low-complexity alternative to CABAC for transform coefficients; (3) A common simple and highly structured variable length coding (VLC) technique for syntax elements not coded by CABAC or CAVLC - Exponential-Golomb Coding.

***MPEG-4 Part 10**

Fast 4x4 Transform

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Forward Transform Matrix

Inverse Transform Matrix

$$\mathbf{T}^{-1} = \begin{bmatrix} 1/4 & 1/4 & 2/10 & 1/10 \\ 1/4 & -1/4 & 1/10 & -2/10 \\ 1/4 & -1/4 & -1/10 & 2/10 \\ 1/4 & 1/4 & -2/10 & -1/10 \end{bmatrix}$$

H.264 Videos

- Fries.mp4 (8 sec, 504kb, 2.1 kb/frame)
- Telephone (8 sec, 332kb, 1.38kb/frame)

The End!

- Next Up: **Project Demos**
- If we have time: Upright Vision With Inverted Spectacles – **A Movie!**
- Enjoy this: Finals Week