Jan-30 Lecture

Video Basics

- Represented by 3 8-bit values
  - RGB: Red, Green, Blue
  - YUV: Luma (Y) and two chrominance: Cr (red) and Cb (blue)
- HDTV as an example
  - 1920 x 1080 x 24 bpp x 30 fps = 1.5 Gbps (60 Hz interlaced)
- Other smaller formats
  - NTSC: 352x240 @ 30fps
  - PAL: Source Input Format (SIF): 352x288 @ 25 fps
  - Quarter Common Intermediate Format (QCIF): 176x144
  - Sub QCIF: 128x96
  - 4CIF: 704x576
  - 16CIF: 1408x1152

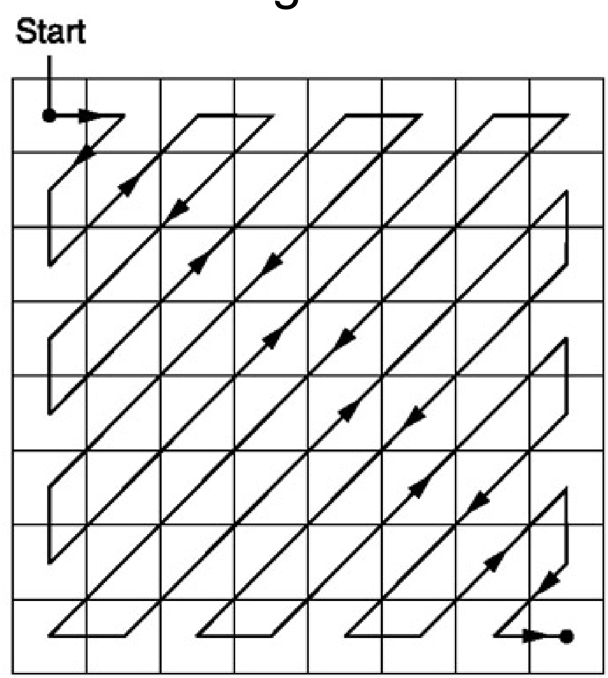
Video Basics

- Three mechanisms used as part of encoding/compression scheme
  1. ***Spatial***: similarities around a given location of a frame
  2. ***Temporal***: similarities around a given location across time
  3. ***Lossy***: eliminate details not visible to the naked eye
     - Downsampling and manipulating the bit stream

- As we discuss different techniques, pay attention to those that increase delay and loss effects:
  – Requiring future data to encode current data
  – Computational complexity
  – Differential encoding

1. Spatial Redundancy

- Focused on similarities within a single image
  - Take advantage of the fact that most images in a video have similar values in nearby positions
- One option is to use differential encoding
  - Similar to delta compression
  - Assume that adjacent values in an image are the same and only encode the difference



2. Temporal Redundancy

- Check for similarities between video frames
  - Can look in the reverse or forward direction
  - Can also use reverse *and* forward direction
- If no other frames exist (single image) or no temporal redundancy is used, the frame is intra-encoded (sometimes called an I-frame)
  - Takes considerably more storage/bandwidth than frames that can take advantage of temporal redundancy

2. Temporal Redundancy

- Predicted images (P-frames) are based on other I-or P- frames

- Bi-directionally predicted images (B-frames) are based on combination of forward and backward prediction

2. Temporal Redundancy

- Predicted images (P-frames) are based on other I-or P- frames
  - Encoder does an expanding ring search to find image components (motion compensation)
  - How far from original location to look corresponds to how much processing is necessary and how much compression is had (key reason encoder is more complex)

- Bi-directionally predicted images (B-frames) are based on combination of forward and backward prediction

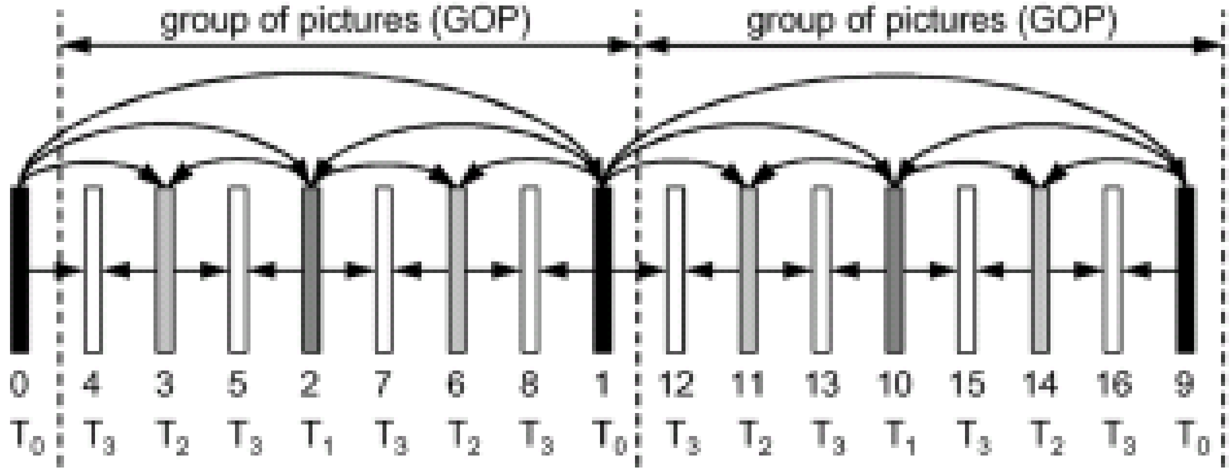
2. Temporal Redundancy

- Predicted images (P-frames) are based on other I-or P- frames
  - Encoder does an expanding ring search to find image components (motion compensation)
  - How far from original location to look corresponds to how much processing is necessary and how much compression is had (key reason encoder is more complex)

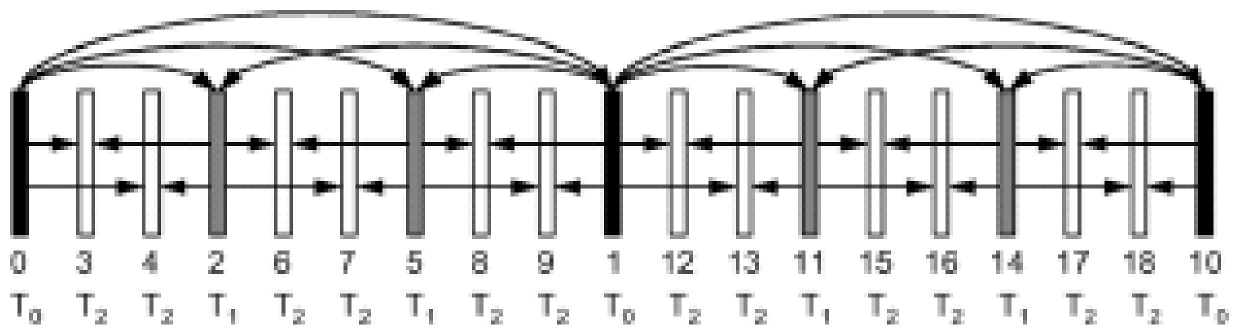
- Bi-directionally predicted images (B-frames) are based on combination of forward and backward prediction
  - If imagine component is in Location A now and Location C in the future. Half way between now and then, it should be in Location B (interpolation).
  - Encode the "error": the difference between predicted Location B and where it actually is

2. Temporal Redundancy

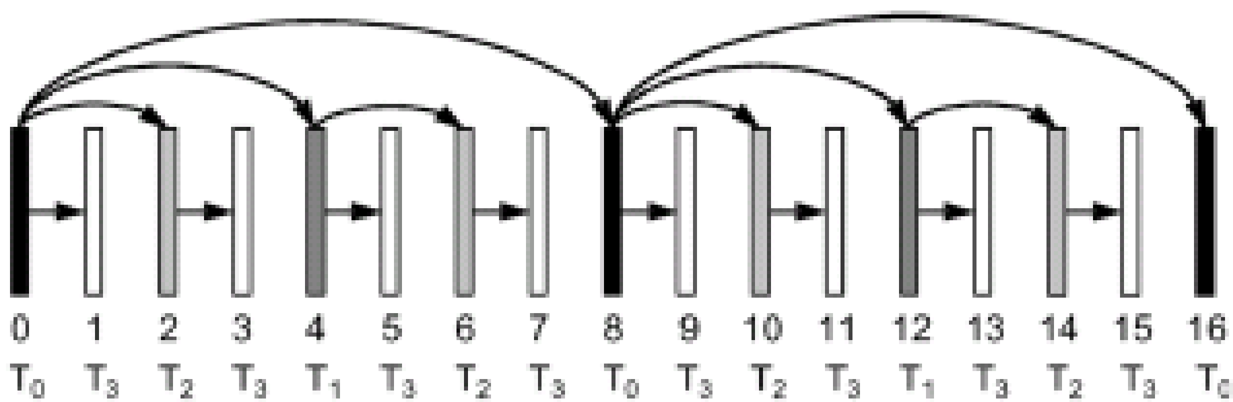
- Thresholds are used for P and B frames
  - If there are enough differences (e.g., a scene change) such that an P or B frame would not result in any less data, then the frame is encoded as an I frame
- Typically use a pattern of I, P, and B frames
  - Ex: I B B P B B P B B … I B B P B B P B B … repeat
  - Could encode all I frames (motion JPEG)
- For real-time video, typically no B frames
  - B frames depend on future frames, can't encode and send until the future I frame is generated (so adds delay)
- For compressed and stored video, different I/P/B patterns can be tried
  - Nice tradeoff between processing and encoding efficiency


group of pictures (GOP)
group of pictures (GOP)
0 4 3 5 2 7 6 8 1 12 11 13 10 15 14 16 9
T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0

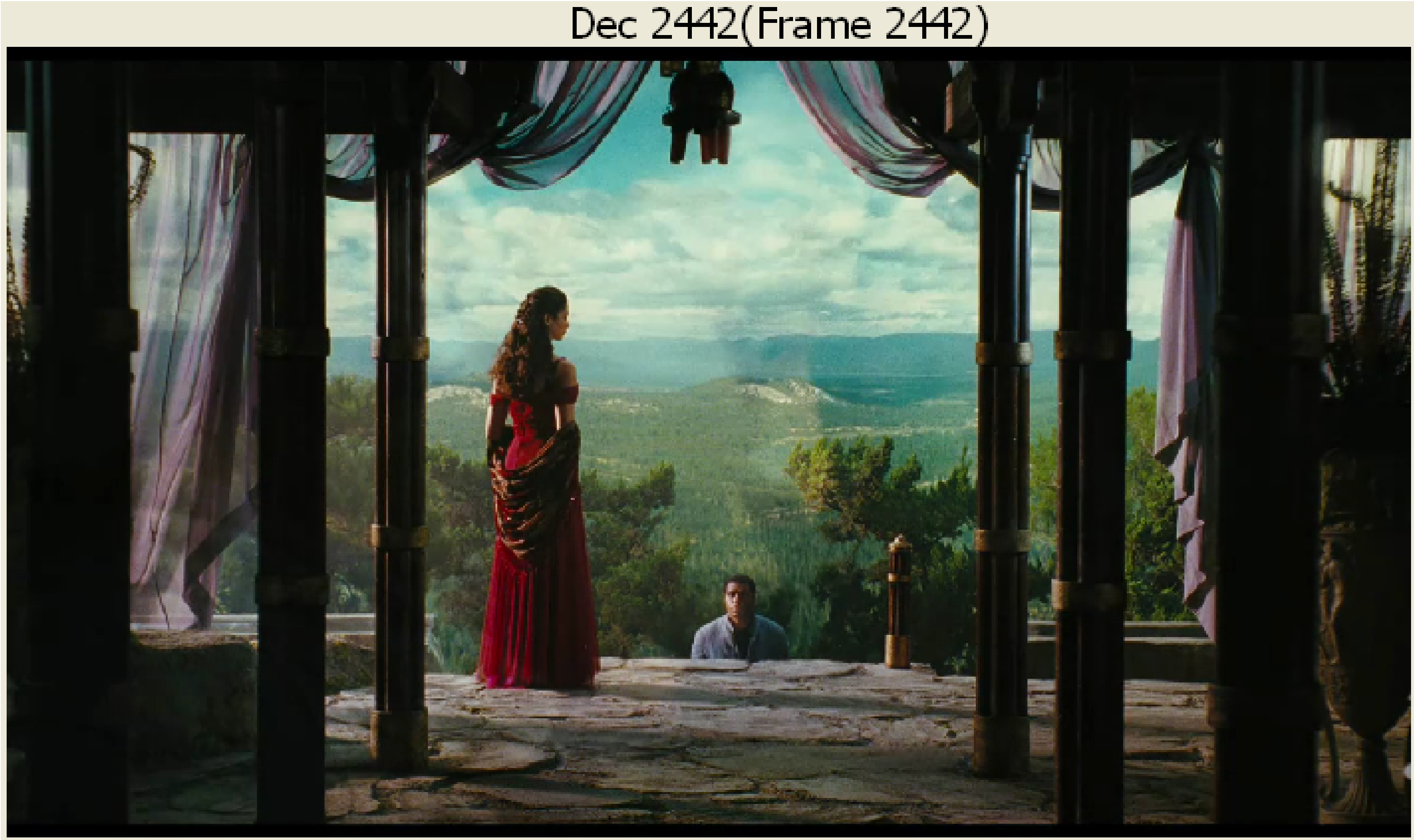
(a)


0 3 4 2 6 7 5 8 9 1 12 13 11 15 16 14 17 18 10
T_0 T_2 T_2 T_1 T_2 T_2 T_1 T_2 T_2 T_0 T_2 T_2 T_1 T_2 T_2 T_1 T_2 T_2 T_0

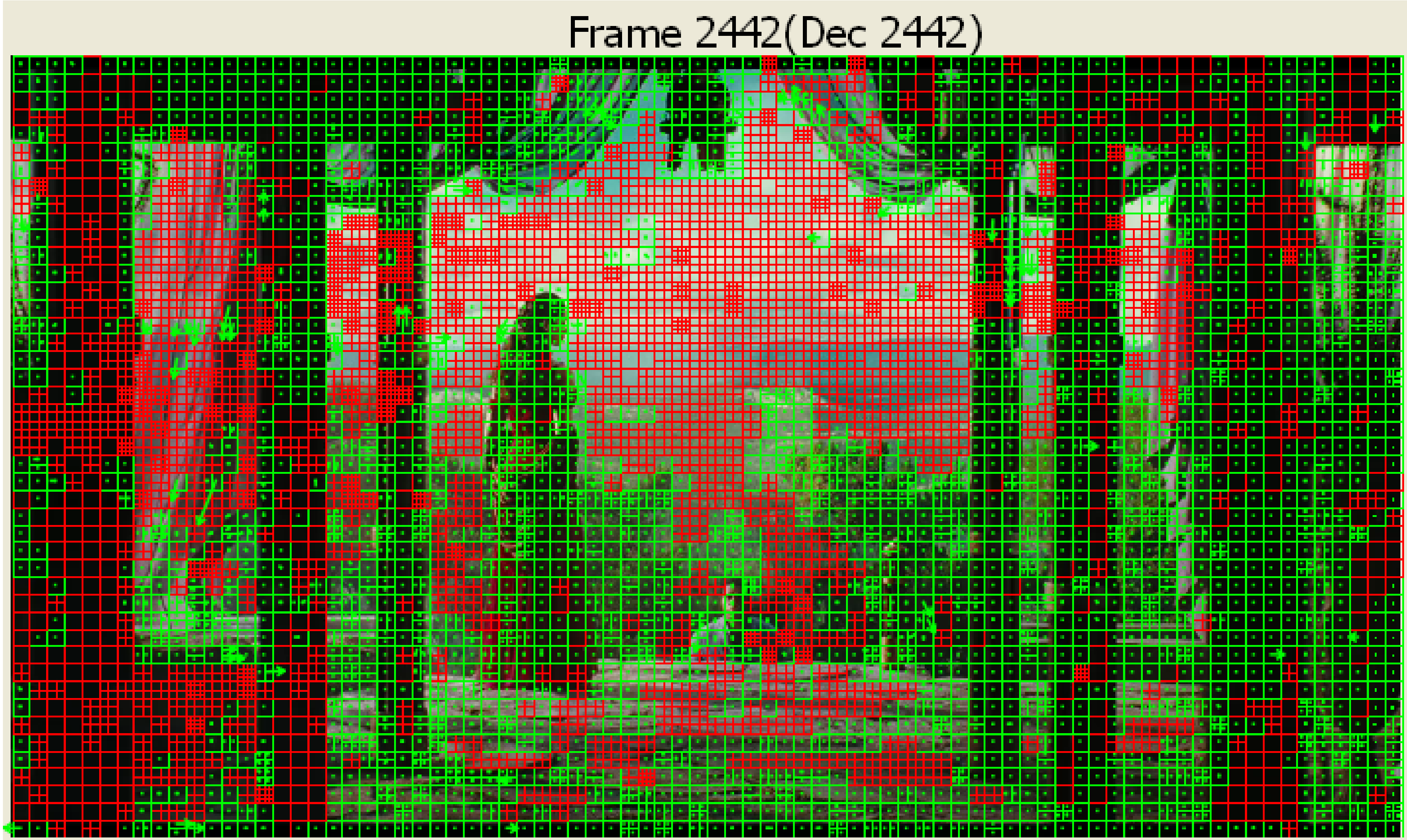
(b)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0

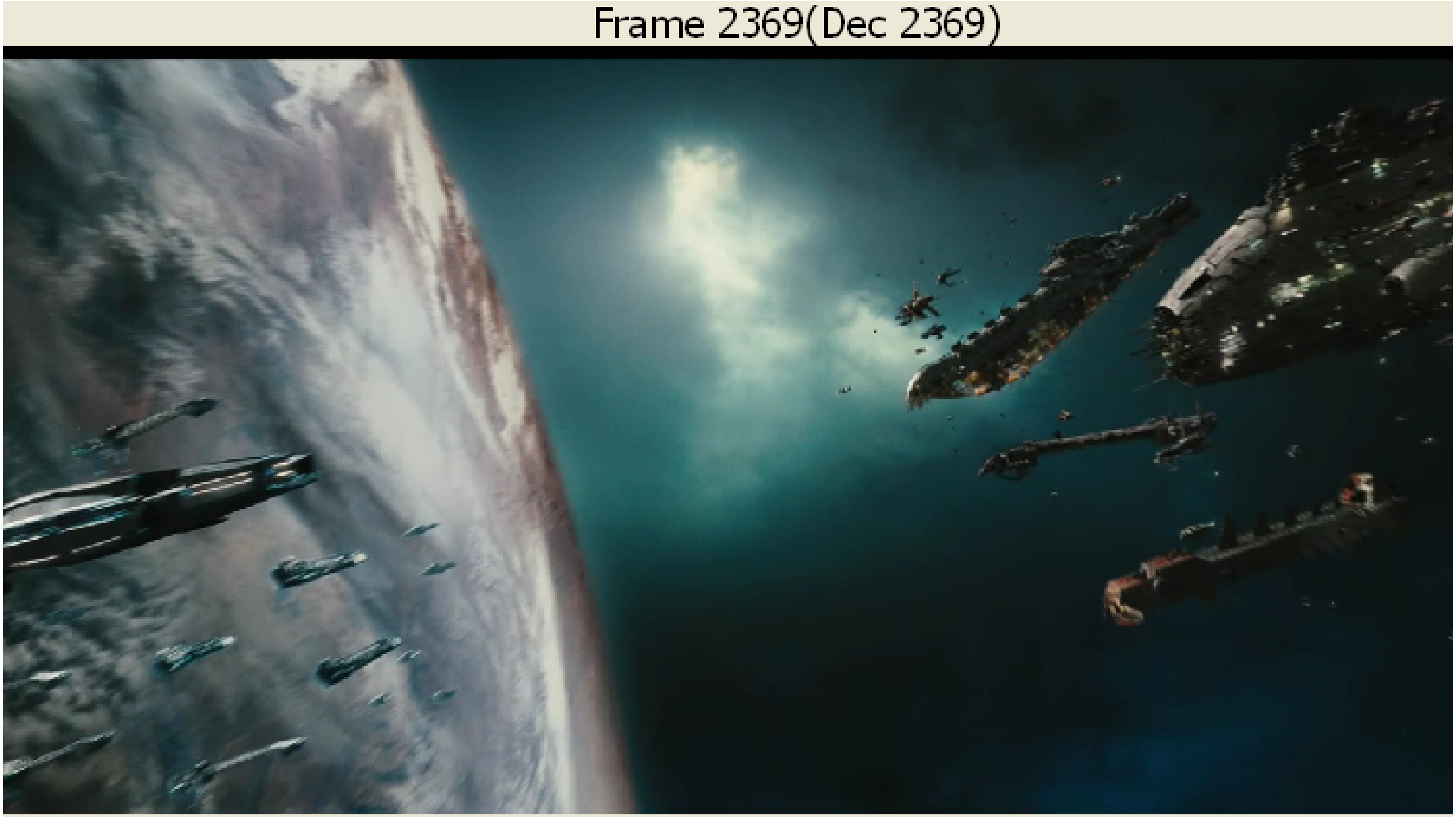
(c)



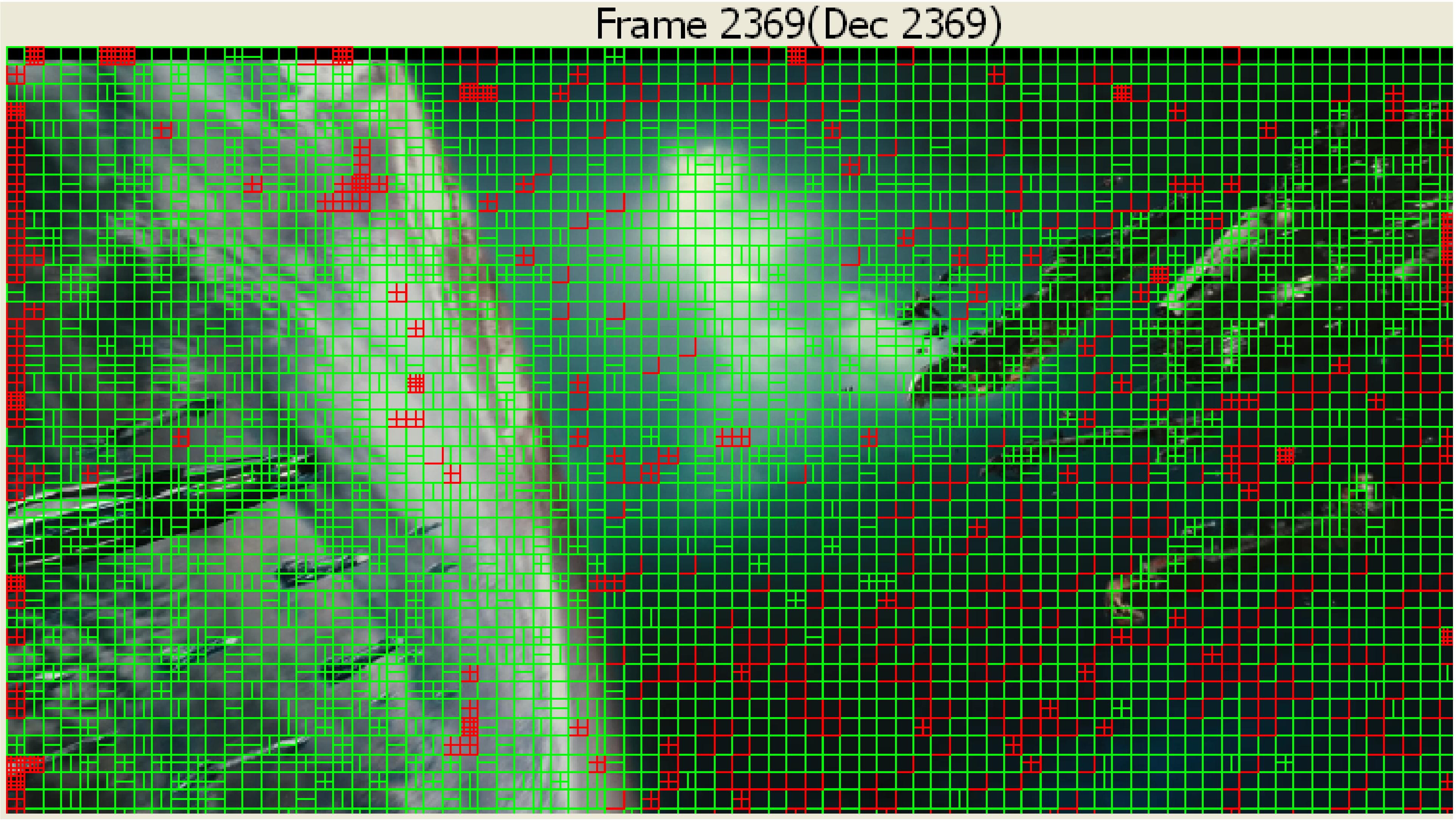
- See Time 1:39 in video



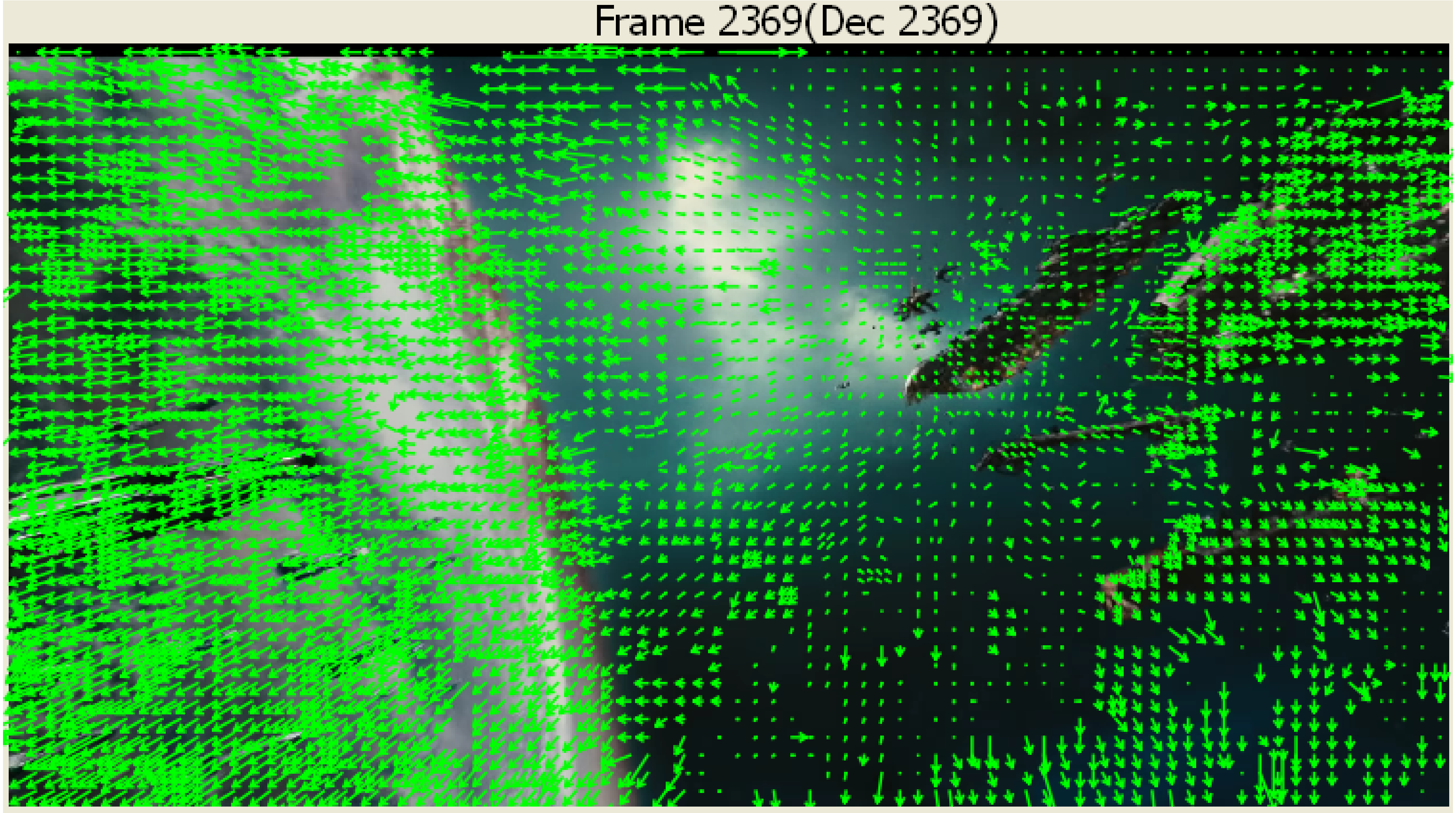
- Green = redundant with other frame
- Red = newly encoded



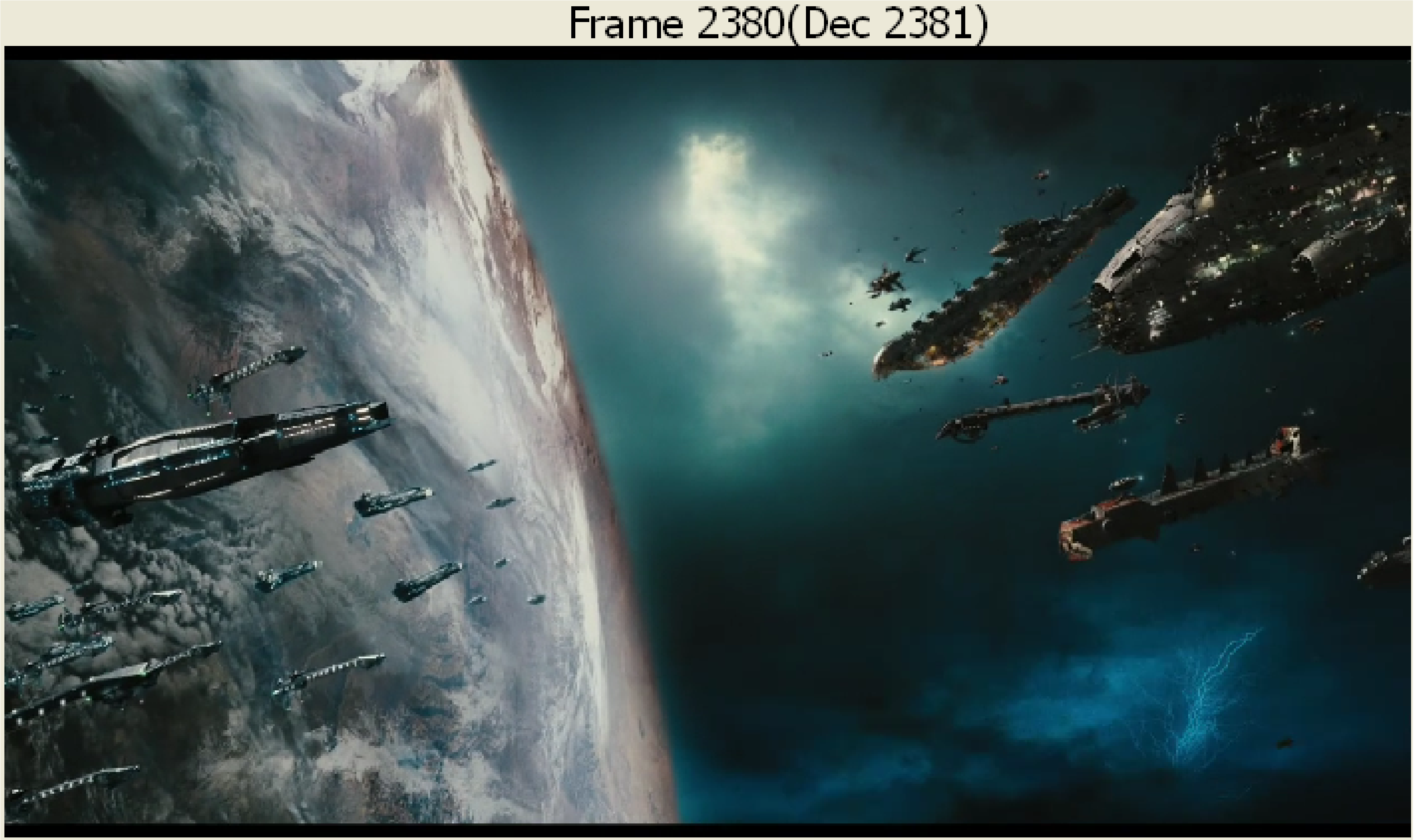
- See Time 1:37 in video



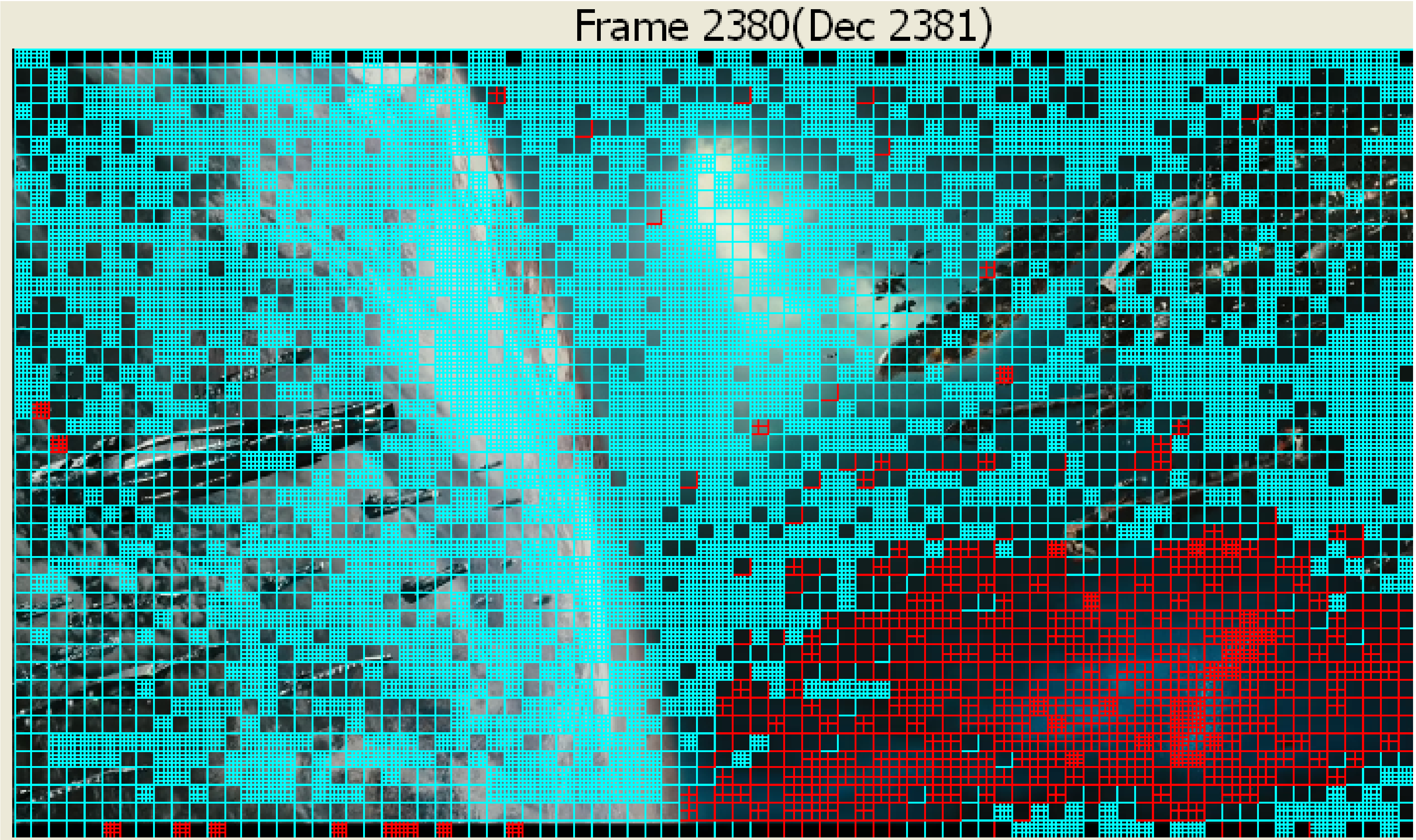
- Green = redundant with other frame
- Red = newly encoded



- Arrows show interpolation



- See Time 1:38 in video



- Blue= redundant with other frame
- Red = newly encoded

3. Lossy Compression

a. Sub-sampling
   - Don’t encode every bit of every block
   - This is step adds loss

b. Apply mathematical transforms
   - Usually something like a Discrete Cosine Transform

c. Quantization
   - Eliminate differences in values
   - This is step adds loss

d. Stream coding
   - Use zig-zag and stream compression

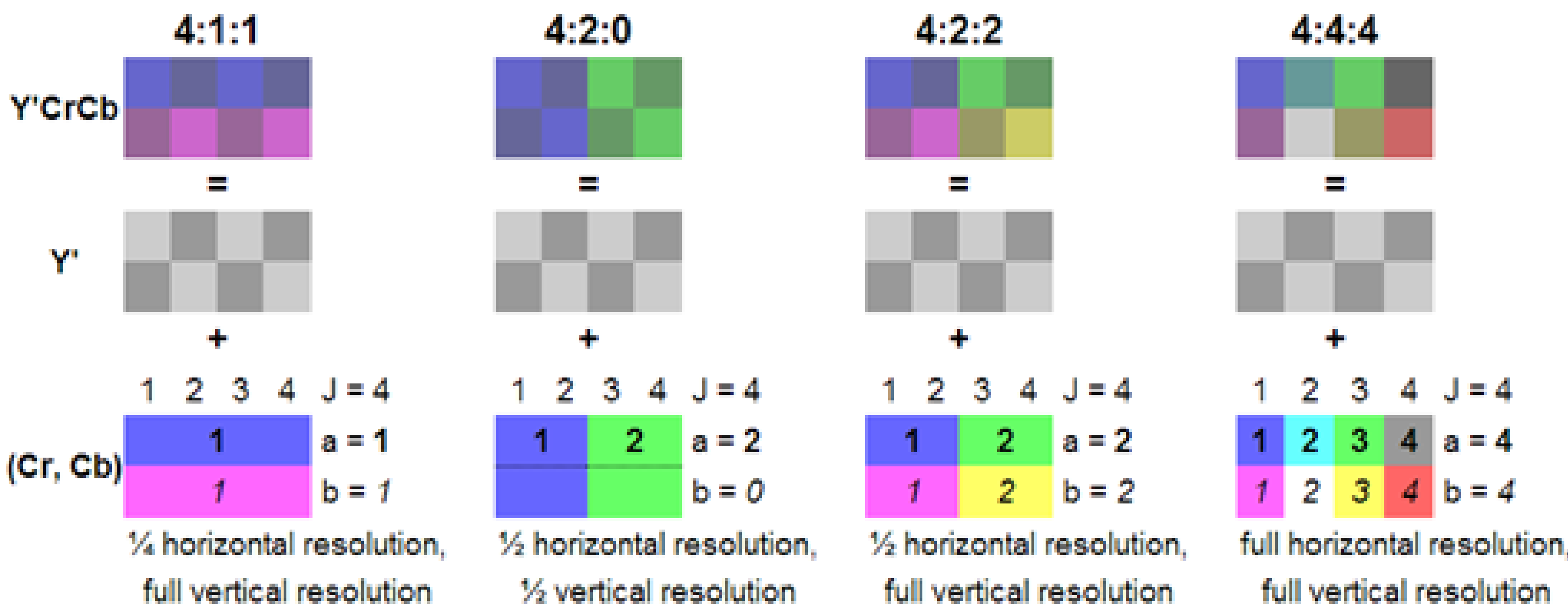
3a. Sub-Sampling

- Eye is most sensitive to changes in luminance, and less sensitive to variations in chrominance

- Start with 16x16 block of pixels (macroblock)
  - Divide in 4 8x8 blocks
  - Full encoding is 12 blocks per macroblock (4 Luma, 4 Cr, and 4 Cb)
  - Syntax is: a:b:c (Luma:Cr:Cb)

3a. Sub-Sampling

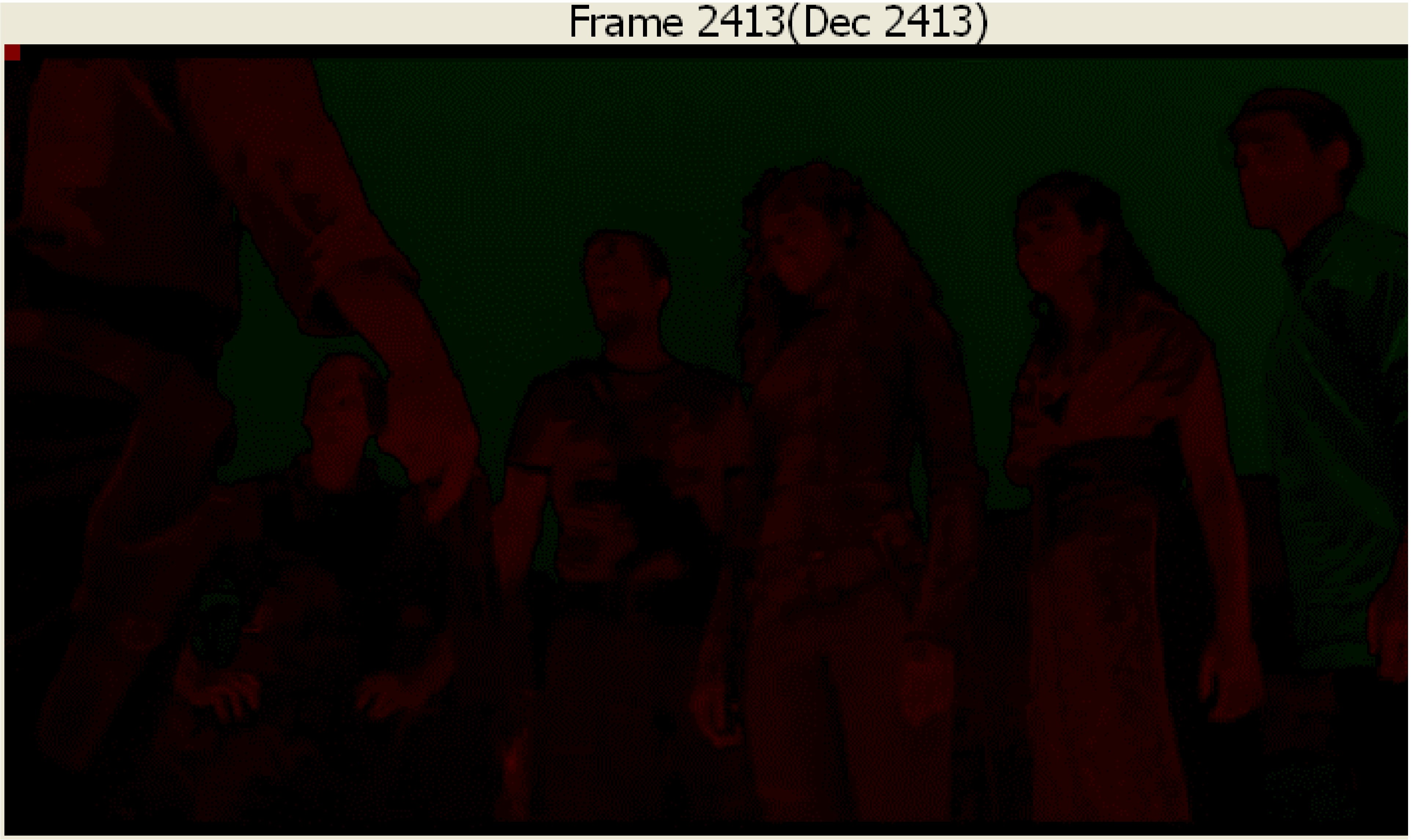
- Eye is most sensitive to changes in luminance, and less sensitive to variations in chrominance
- Start with 16x16 block of pixels (macroblock)
  - Divide in 4 8x8 blocks
  - Full encoding is 12 blocks per macroblock (4 Luma, 4 Cr, and 4 Cb)
  - Syntax is: a:b:c (Luma:Cr:Cb)



- See Time 1:40 in video

- Y (luminance--brightness) values only

- U (chrominance #1) values only

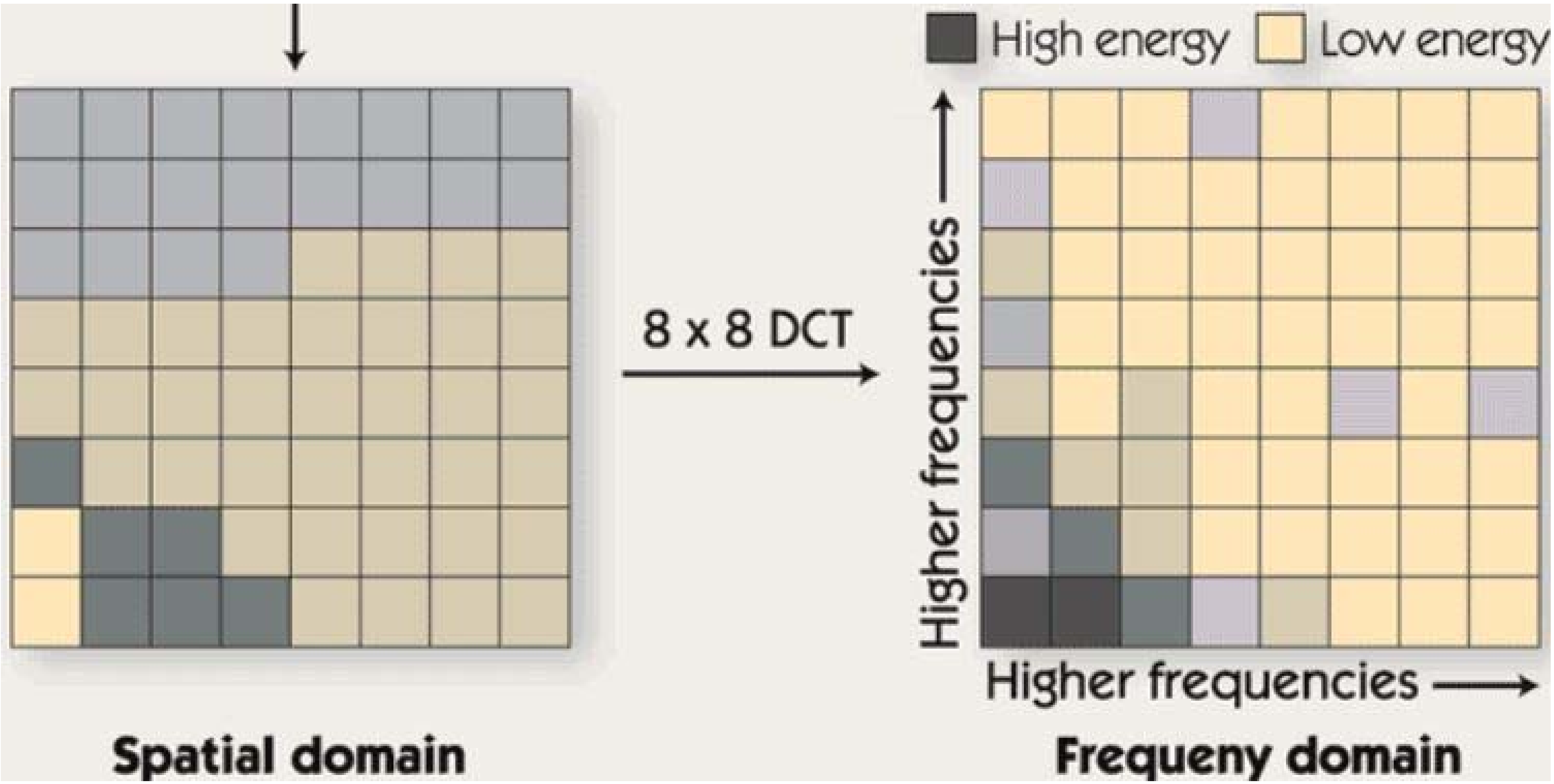


- V (chrominance #2) values only

3b. Transform Coding

- Transform coding is used to convert spatial image pixel values to transform coefficient values
  - No information is lost, the number of coefficients produced is equal to the number of pixels transformed
- The result is that most of the "energy" in the image will be contained in a few large transform coefficients
  - Generally, only a few coefficients will contain most of the energy in a block
  - Smaller coefficients can be coarsely quantized or deleted without doing visible damage to the reproduced image

3b. Transform Coding



3b. Transform Coding

- Many types of transforms have been tried for picture coding
  - Fourier, Karhonen-Loeve, Walsh-Hadamard, lapped orthogonal, discrete cosine, and wavelets
- Goals
  - The most concentration of energy
  - Least number of artifacts

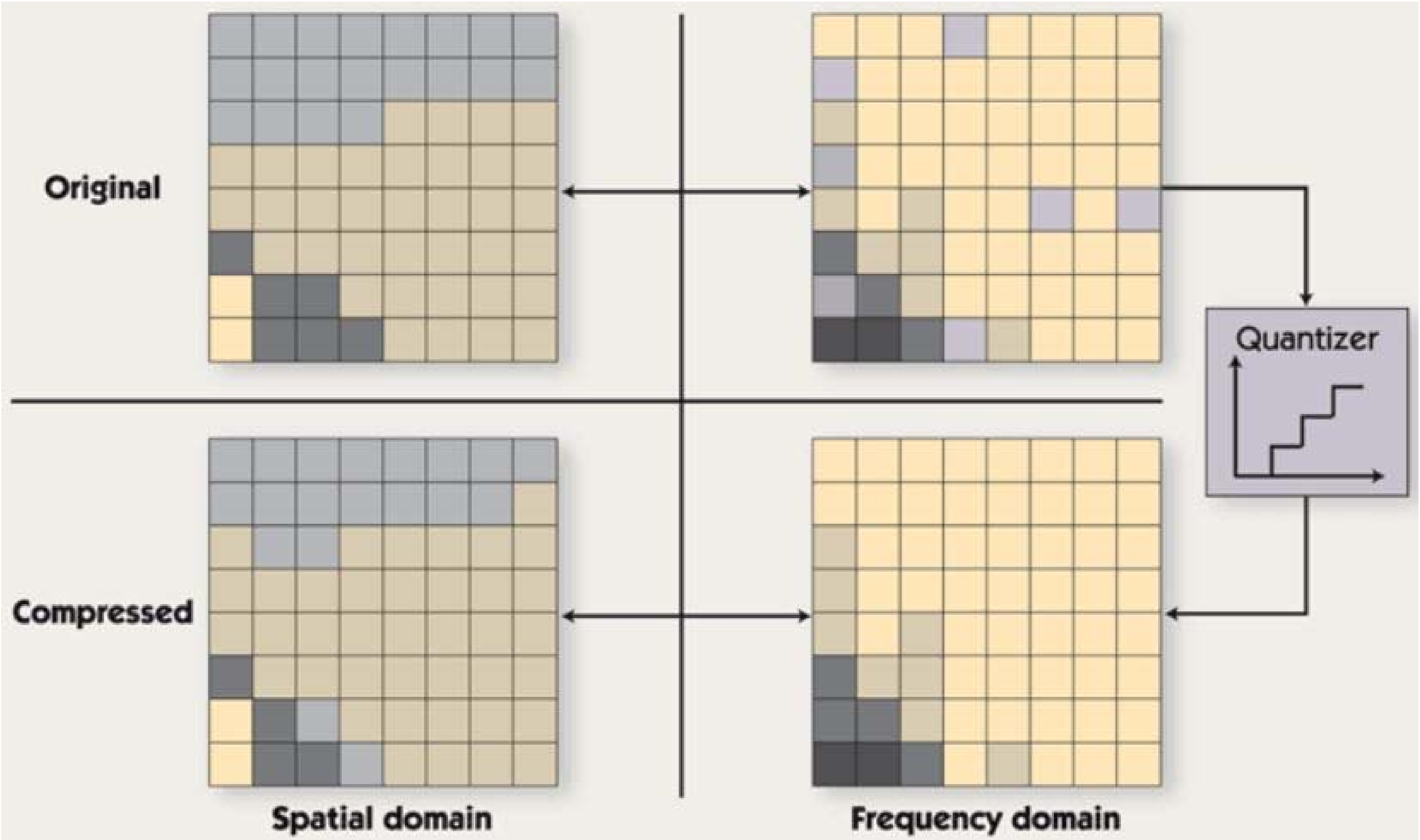
3c. Quantization

- The level of quantization provides excellent tradeoff between quality and level of compression
  - More quantization means more compression which means less bandwidth but more artifacts

- Quantization can be adjusted dynamically
  - **Constant Bit Rate (CBR)**: same amount of bandwidth no matter the amount of energy/action in a picture
  - ***Variable Bit Rate (VBR)***: bandwidth requirements vary based on complexity and motion in video

- Use of quantization is the source of noise/error in a compressed stream (different than network data loss)

3c. Error/Noise



3c. Error/Noise

- *Coding error*: the difference between the source picture and the reproduced picture
  - Just like for audio
- Coding error is measured as the root-mean-square between the two values
  - A common metric for evaluating the performance of an encoding system

3d. Stream Coding

- An example block of 8x8 DCT samples:

12	34	0	54	0	0	0	0
87	0	0	12	0	0	0	0
16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

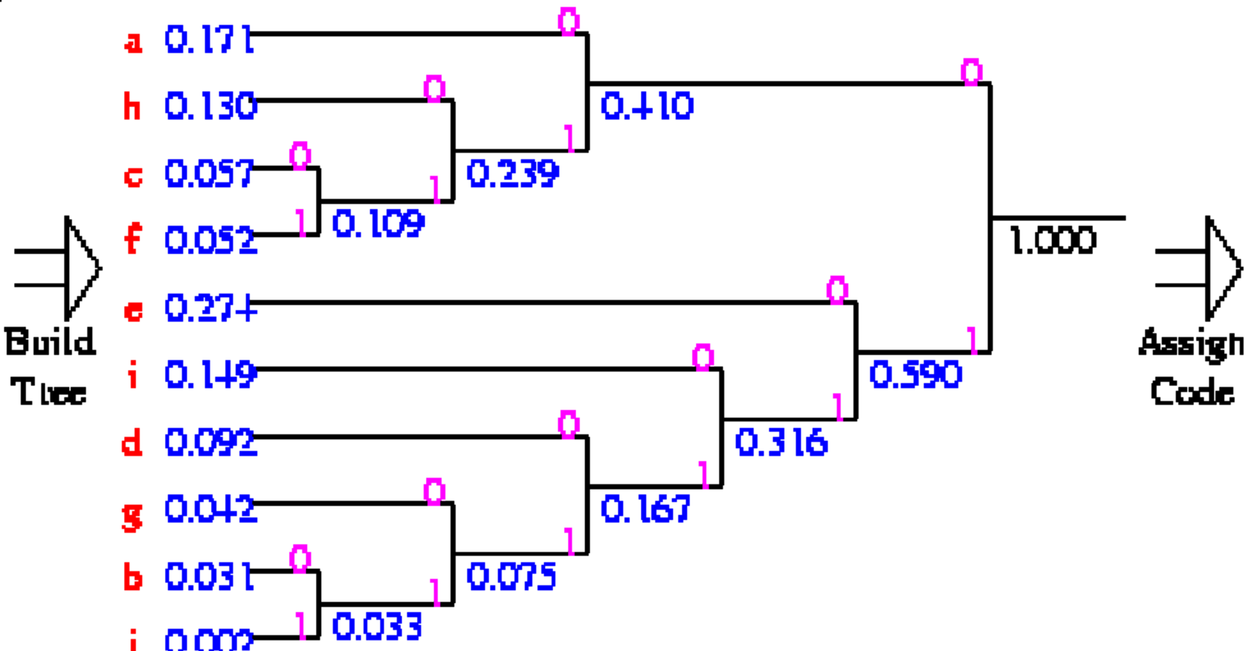
- First do zig-zag sequencing:
  - 12 34 87 16 0 0 54 0 0 0 0 0 12 0 0 0 0 0 .....

3d. Stream Coding

- Zig-zag:
  - 12 34 87 16 0 0 54 0 0 0 0 0 0 12 0 0 0 0 0 0 .....
- Then apply quantization:
  - 12 36 88 16 0 0 56 0 0 0 0 0 0 12 0 0 0 0 0 0 .....
- Then apply run-length coding
  - Instead of long sequence of zeros, replace with “sequence of X zeros”
- Then apply Huffman coding
  - Most common sequence of bits is represented by shortest code

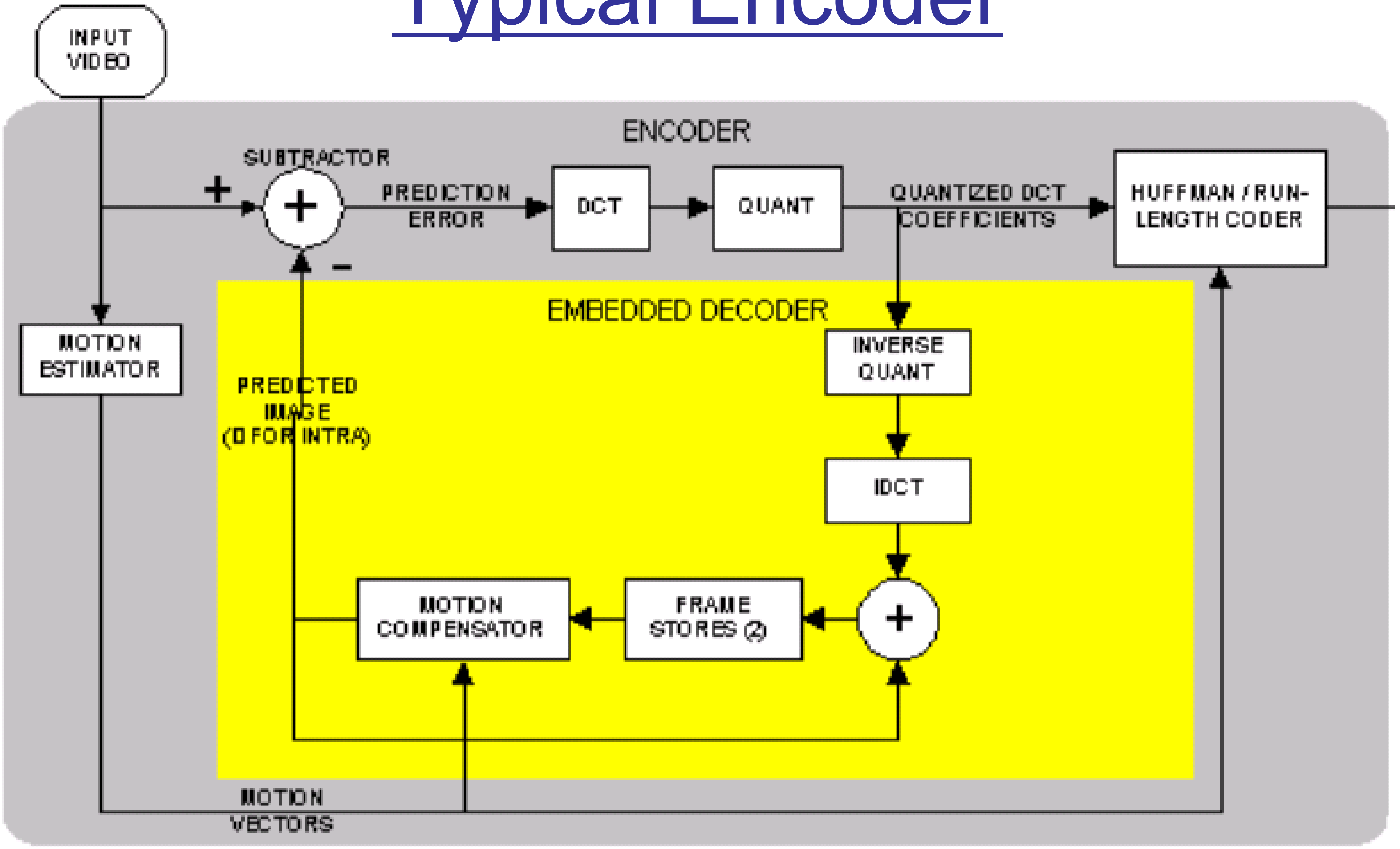
3d. Stream Coding

X	p(X)
a	0.171
b	0.031
c	0.057
d	0.092
e	0.274
f	0.052
g	0.042
h	0.130
i	0.149
j	0.002



X	Code
a	0 0
b	1 1 1 1 1 0
c	0 1 1 0
d	1 1 1 0
e	1 0
f	0 1 1 1
g	1 1 1 1 0
h	0 1 0
i	1 1 0
j	1 1 1 1 1 1

Typical Encoder



Standards

- MPEG-1 (1993)
  - Designed up to 1.5 Mbps
  - Standard CD-ROM, NTSC video quality
- MPEG-2 (1995)
  - Designed for between 1.5 and 15 Mbps
  - Standard for DVD, HDTV
- MPEG-4 (1999)
  - Object-based compression
- MPEG-7 (2002)
  - Provides framework for adding descriptive information about video contents: uses XML to store meta-data
- MPEG-21(2001)
  - Adds digital rights/permissions/restrictions

Other Standards

- MPEG came out of ISO
- Also CCITT (which became ITU)
  - Early on, principally designed encoding for low-bit rate video conferencing (Ex: H-261, H-263, H-264, etc.)
  - Typically use the same components (e.g., temporal, spatial, etc.) in the encoding scheme
- Typically there are separate standards for audio and video and then for a combination of the two

MPEG-1

- Consists of 5 parts:
  - Systems (storage and synchronization of video, audio, and other data together)
  - Video (compressed video content)
  - Audio (compressed audio content)
    - 3 different layers, the third is most commonly used (MP3)
  - Conformance testing (testing the correctness of implementations of the standard)
  - Reference software (example software showing how to encode and decode according to the standard)

MPEG-2

- Pictures are either “frame picture” or “field picture”
  - “frame picture”: complete frame
  - “field picture”: half of interlaced frame
- Has a header
  - Picture type (I, P, B)
  - Temporal reference information
  - Motion vector search range
  - Optional user data
- Sequence is a group of Groups of Pictures (GOPs)
  - Series of I, P, and B frames
- Further divided: Slice is a row of Macroblocks is group of Blocks

Other Standards

- Almost all other coding standards have the same basic elements

- Differences include:
  - Different techniques within each function
    - Ex: a different transform
  - Different "stream" structure (e.g., headers and data org)
    - lossy transmission
    - asymmetric coding: processing-intensive encode but easy decode
    - battery efficient decode
    - limited capacity transmission links
  - Some include special features
    - Ex: Google's VP8 has a golden frame

Network Transmission

- Only some standards are designed for transmission
  - What happens when part of an I frame or P frame is lost?
  - What happens when header information is lost?

- There are steps that can be taken to improve resiliency
  - Basic idea is to have fewer critical elements
    - Not have multiple/many frames rely on same portions of the stream
  - Increased redundancy means less susceptible to loss, but at the tradeoff of sending more information

Network Considerations

- Most video now is TCP-based
  - Even for real-time and interactive streams
  - So use of UDP-based streaming is fairly limited

- Still, even with TCP, the goal is to avoid congestion and TCP backoffs
  - Catastrophic if TCP is used for interactive
  - Problematic if TCP is used for real-time

- A difference from audio is that video can consume a significant amount of bandwidth and congestion becomes a real concern

Congestion Solutions

- Rely on the user to choose the right quality
- Develop a dynamic solution that monitors loss rates and changes the quality in response to congestion (or the increasing likelihood of congestion)
  - Monitor throughput and increase/decrease quality
  - Increase/decrease frame rate
  - Increase/decrease quantization
  - Increase/decrease frame size
  - Use layered encoding

Scalable Video Coding

- Create set of *dependent* layers such that each additional layer adds detail and clarity
- Very useful for creating and sending a video stream to lots of people simultaneously but who have different end-to-end throughput levels
  - Adds complexity and inefficiency (more overhead to encode dependent layers)
- Alternative is to send separately encoded streams
  - Same stream encoded at different levels—requires switching to different stream

H.264 Scalable Video Coding

- One standardized solution uses H.264, an ITU standard for video compression
- Basic idea is how to do scalable video coding
  - Additional B frames can be added
  - Additional B frames are dependent on the higher level B frames directly on either side of the current level

Modern Video Streaming

- M. Tourad Diallo, H. Moustafa, H. Afifi, N. Marechal, “Adaptation of Audiovisual Contents and Their Delivery Means,” Communications of the ACM, vol. 56, no. 11, pp. 86-93, November 2013.