Digital Image Processing

Image and Video Compression

Topics

- Image Compression
  - Run-length
  - Huffman Coding
  - Arithmetic Coding
  - JPEG
- Video Compression
  - Mpeg coding
- Multi-resolution Image Processing
  - Wavelet Transform
  - Image Compression using DWT

Data Redundancy

- Redundancy in information theory is the number of bits used to transmit a message minus the number of bits of actual information in the message.

- Informally, it is the amount of wasted "space" used to transmit certain data

Data Compression

- Data compression or source coding is the process of encoding information using fewer bits (or other information-bearing units) than an un-encoded representation would use through use of specific encoding schemes.

Image Compression

Image Compression

- Importance of Image Compression
  - Typical image resolution/depth
    - 1024 x 1024 x 24
    - Typical image size in bytes: 3 MB
  - Typical video rate/resolution/depth
    - 30 fps, 720×576 , 24 bits
    - Typical video size for 1 minute: More than 2 GB

Compression Methods

- Lossless Methods
- Lossy Methods

Run Length Coding

- In this method runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count.
- Useful in relatively simple graphic images such as icons, line drawings, and animations.
- Example,

  WWWWWWWWWWWWBWWWWWWWWWWWW
  WWBBBWWWWWWWWWWWWWWWWWWWW
  WWWWWWBWWWWWWWWWWWWWWWWW

  is stored as : 12W1B12W3B24W1B14W

Entropy Coding

- Entropy coding is a lossless coding method.
- Entropy coding creates variable length codeword. The length of each codeword is approximately proportional to the negative logarithm of its probability.
- Huffman Coding

  Huffman coding is an entropy encoding algorithm used for lossless data compression.
- The term ***Entropy Coding*** refers to the use of a variable-length code table for encoding a source symbol

Example of Variable Length Coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

Huffman Coding Example

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Huffman Coding Example

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Arithmetic Coding

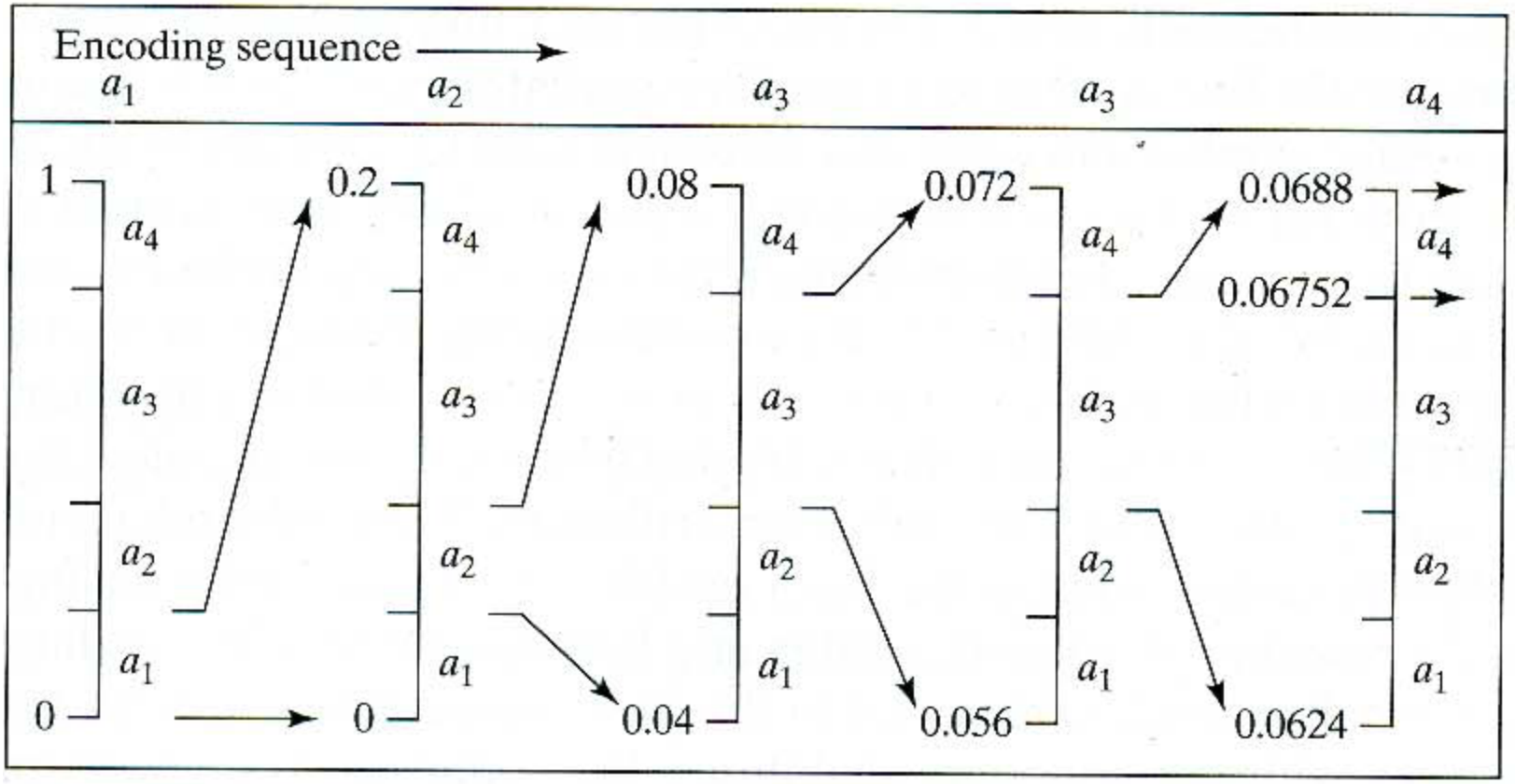
- The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0

- This single number can be uniquely decoded to create the exact stream of symbols that went into its construction.

Example: $a_1a_2a_3a_3a_4$

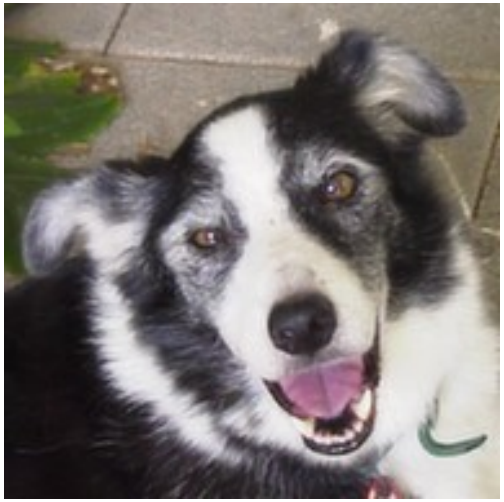
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Example


Encoding sequence
a_1
a_2
a_3
a_3
a_4
1
0
0.2
0
0.08
0.04
0.072
0.056
0.0688
0.06752
0.0624
a_4
a_3
a_2
a_1

Lossy Compressions

- A lossy compression is a data compression method which discards (loses) some of the data, in order to achieve a high compression rate, with the result that decompressing the data yields content that is different from the original one.



JPEG Standard

- Steps followed in JPEG
  - Color space transformation
  - Down-sampling
  - Block splitting
  - Discrete cosine transform
  - Quantization
  - Run length and Entropy coding

Color space transformation

- First, the image should be converted from RGB into YCbCr color space
- The compression is more efficient as the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel, more closely representing the human visual system.

Down Sampling

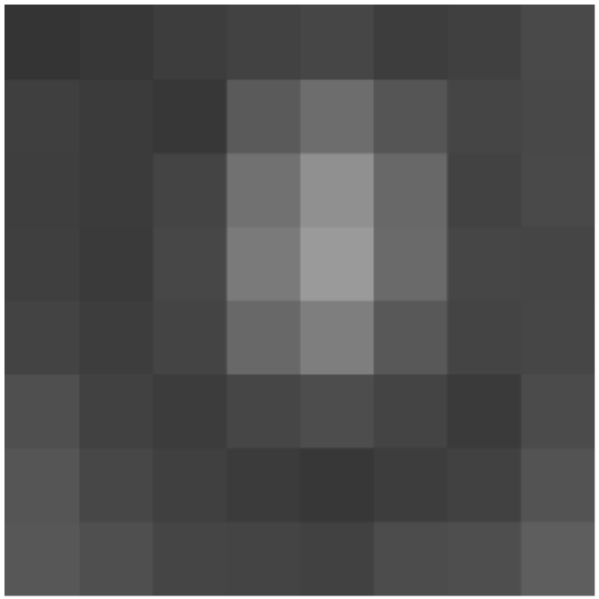
- Due to the densities of color- and brightness sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y component) than in the color of an image (the Cb and Cr components).
- Down-sample Cb and Cr components only 4:2:0

Splitting the Frame into Blocks

- After sub-sampling, each channel must be split into 8×8 blocks of pixels.

Discrete cosine transform

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



Discrete Cosine Transform

$$G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right]$$

$$\alpha_p(n) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } n = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$

Result of DCT

$$
\begin{matrix} & & & & u & & & & \\ & & & & \longrightarrow & & & & \end{matrix}
$$

$$
\begin{bmatrix}
-415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\
4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\
-47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\
-49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\
12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\
-8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\
-1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\
0 & 0 & -1 & -4 & -1 & 0 & 1 & 2
\end{bmatrix}
\quad \downarrow v
$$

Quantization

- Quantization is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer

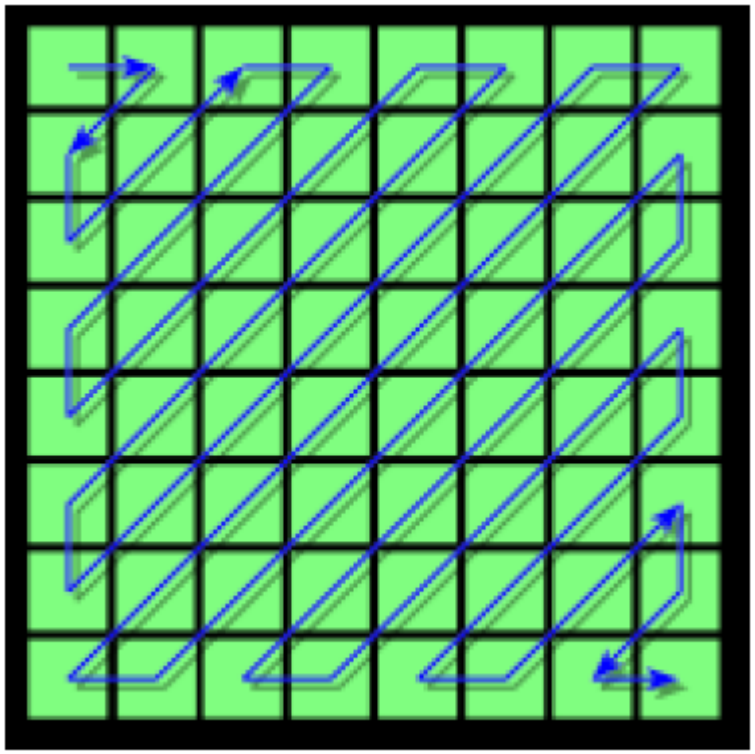
Sample Quantization Table

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization Results

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zig-Zag Scanning and Run Length Coding



Zig-Zag Scanning

−26							
−3	0						
−3	−2	−6					
2	−4	1	−4				
1	1	5	1	2			
−1	1	−1	2	0	0		
0	0	0	−1	−1	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							

Variable length codes

- Variable length codes are generated for the code words of the zig-zag scan result.
- Trailing zeros are truncated by inserting an EOB symbol after the last non-zero value.

Typical compression ratios

- The compression ration depends on the quantization factors used.
- Typical ratio is 4:1

Video Compression

- Video may be defined as a sequence of still images taken at very short time intervals.
- Each image of a video is called a frame
- The number of frames per second is called frame rate or time resolution of the video

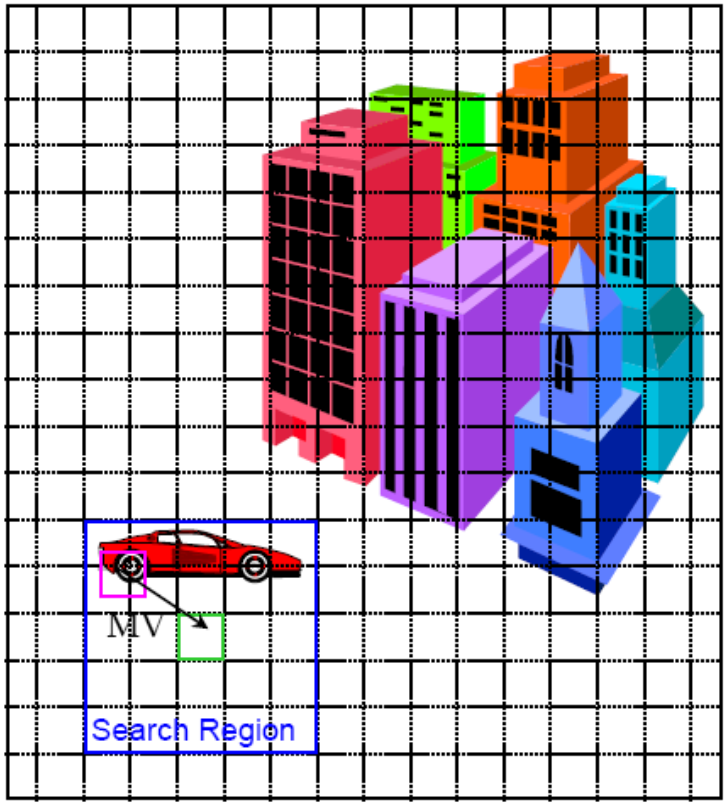
Temporal Redundancy

- Short time interval between the frames of a video implies a high degree of similarity between the consecutive frames.
- This similarity can be used to further compressing the video data.
- The redundancy resulting from the similarity of the frames of a video is called temporal redundancy

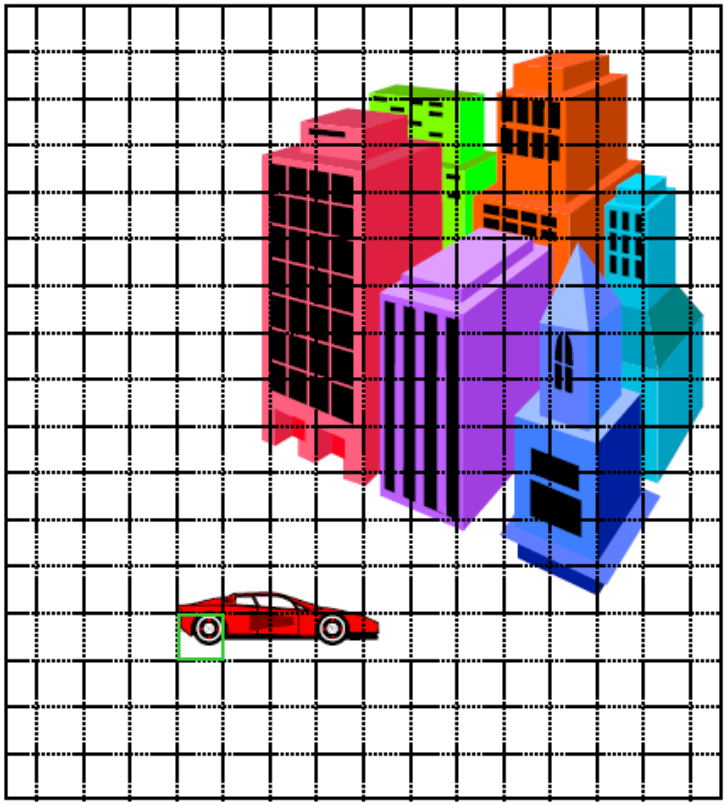
Motion Estimation

- For each frame block, the most similar location in its previous frame is found.
- To eliminate temporal redundancy, only the difference with the most similar area is encoded.
- The location of the most similar area in the previous frame is indicated using a Motion Vector

Motion Estimation



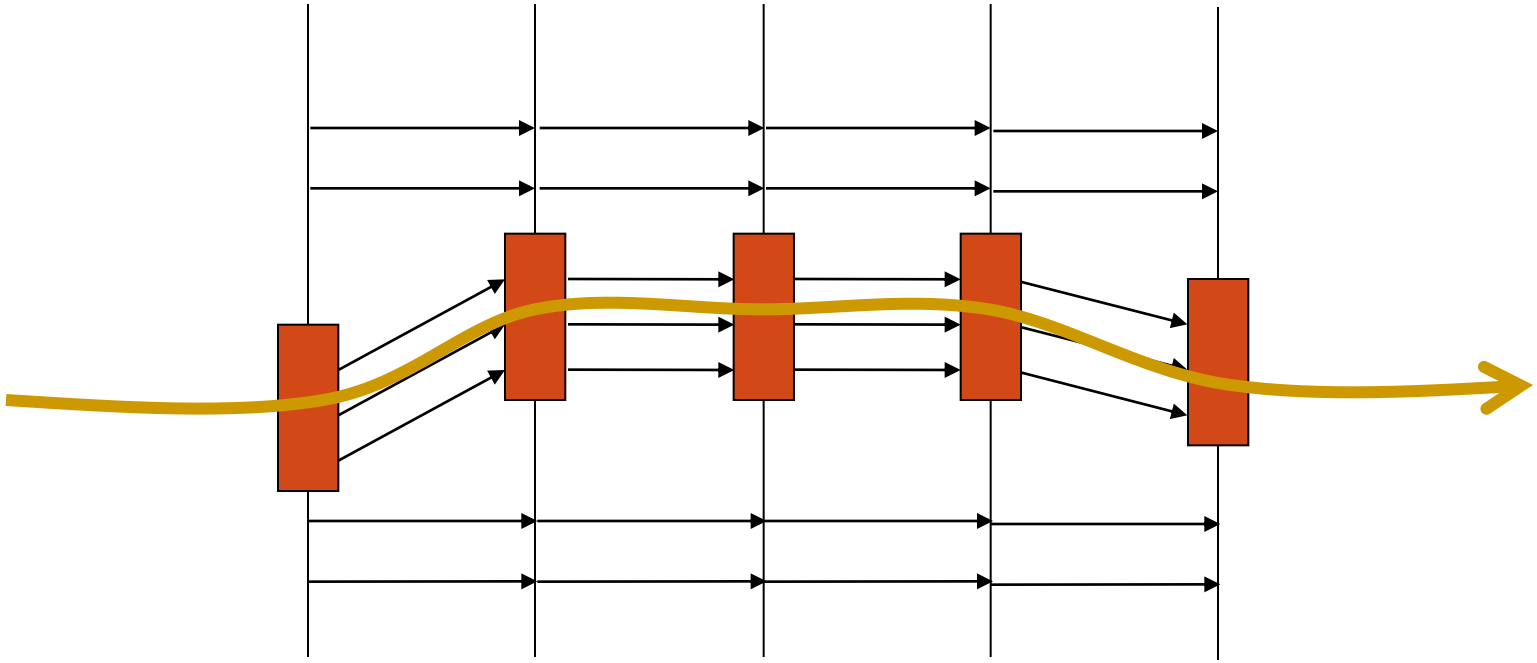
Frame *t-1*
(Reference Frame)



Frame *t*
(Predicted frame)

MCTF

- MCTF = Motion Compensated Temporal Filtering



Similarity Measurement

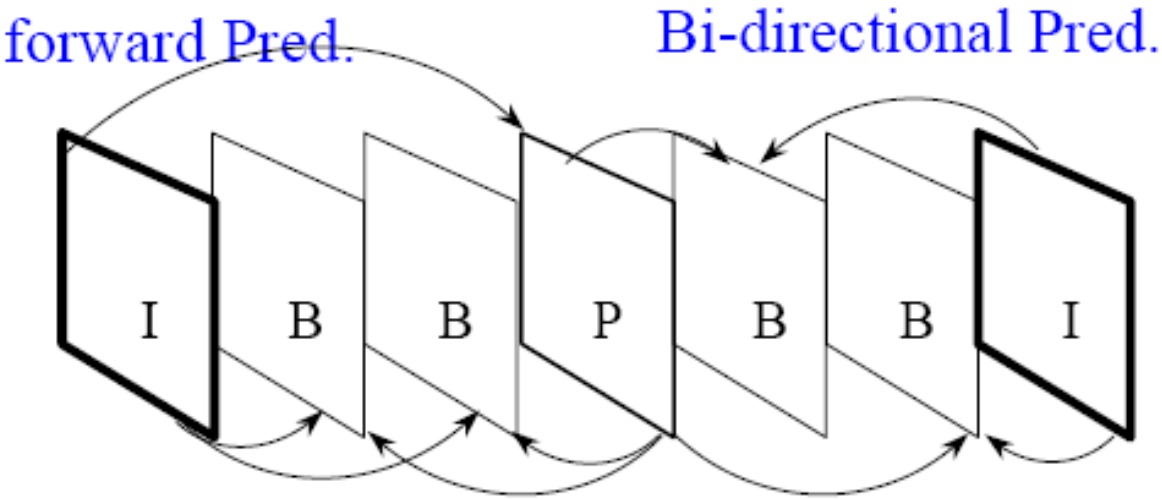
- Sum of Absolute Difference (SAD) is generally used for similarity measurement.

- SAD is defined as:

$$SAD = \sum_i \sum_j | B_i - R_j |$$

where B is the block and R is the reference image

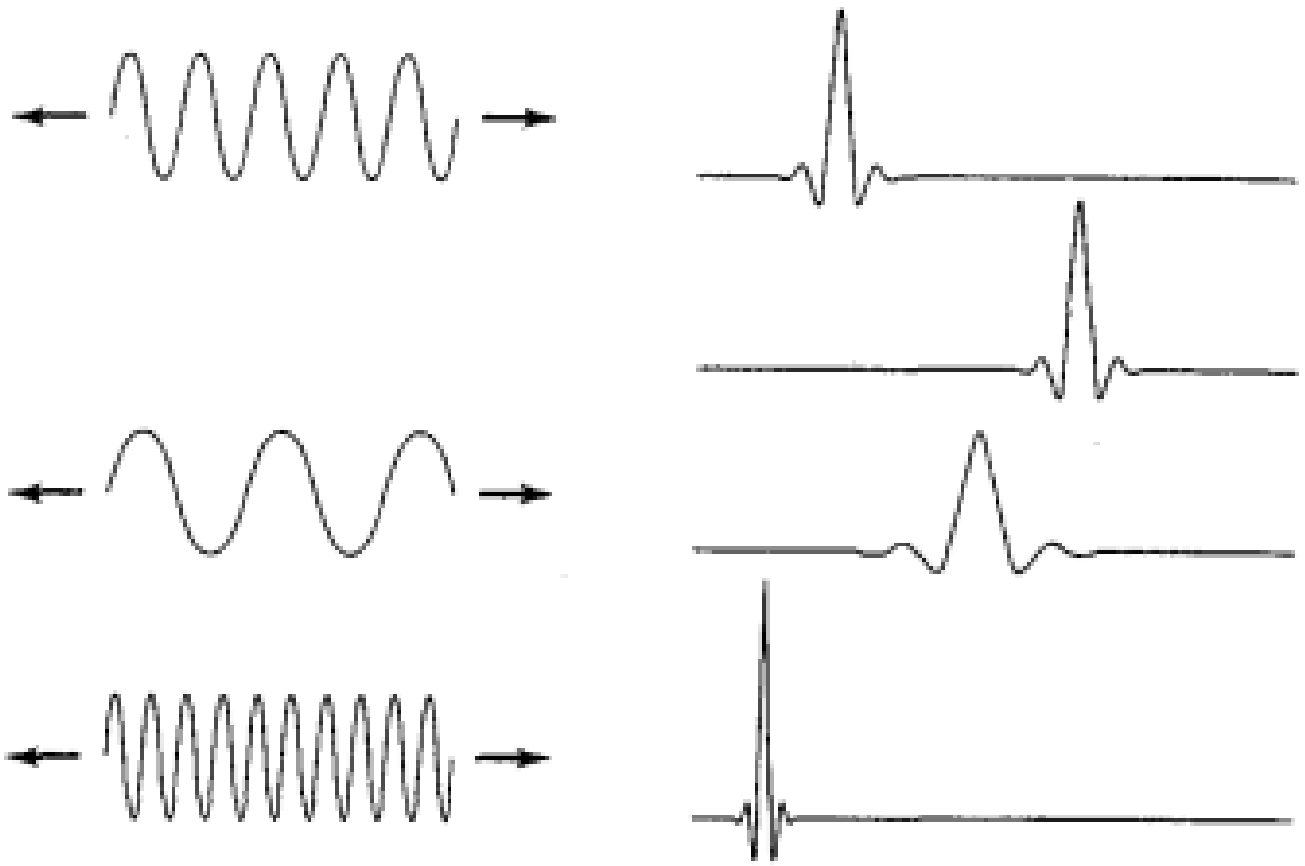
Group of Pictures



Discrete Wavelet Transform

- Unlike the discrete Fourier transform, which can be completely defined by two transform equations using two kernel functions, Discrete Wavelet Transform refers to a class of transforms that differ not only in the transformation kernels but also in the fundamental nature of those functions

Fourier and Wavelet expansion Functions



DWT Properties

- Separability, Scalability, and Translatability

$$\psi^H(x, y) = \psi(x)\varphi(y)$$
$$\psi^V(x, y) = \varphi(x)\psi(y)$$
$$\psi^D(x, y) = \psi(x)\psi(y)$$

$$\varphi_{j,k}(x) = 2^{j/2}\varphi(2^j x - k)$$
$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$$

DWT Properties

- Multiresolution

Haar Transform

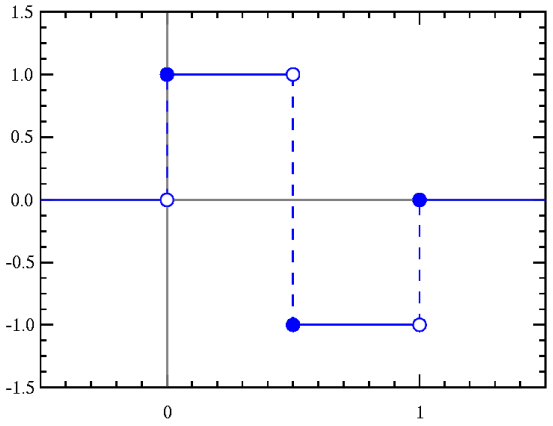
- **Haar wavelet** is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis

The Haar wavelet's mother wavelet function $\psi(t)$ can be described as

$$\psi(t) = \begin{cases} 1 & 0 \le t < 1/2, \\ -1 & 1/2 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Its scaling function $\phi(t)$ can be described as

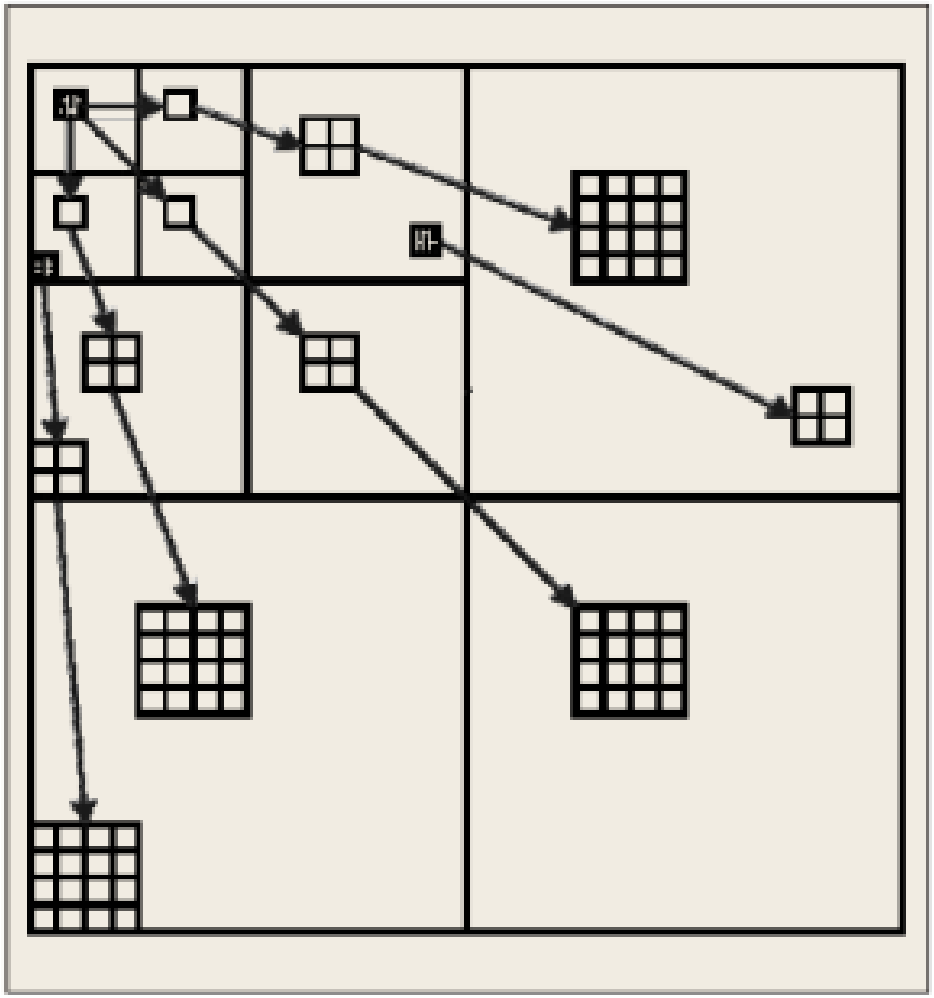
$$\phi(t) = \begin{cases} 1 & 0 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

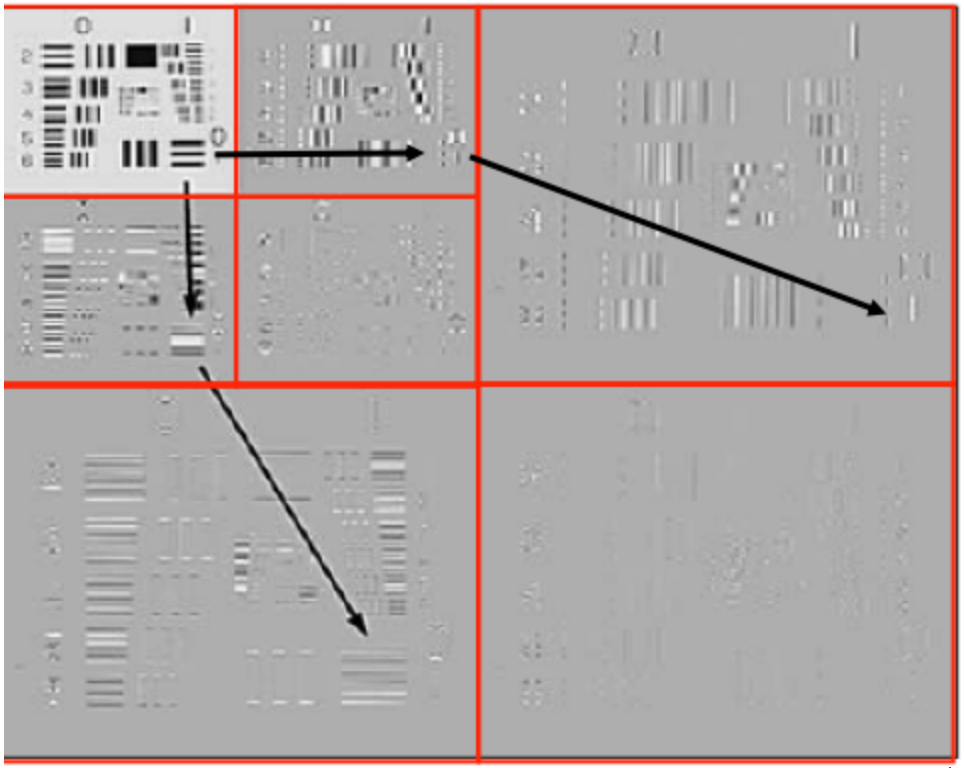


Spatial Zero-Tree Wavelet (EZW)

- Multi-resolution property of DWT gives rise to self-similarity property
- The self-similarity property can be used to estimate the location of significant coefficients
- Non-significant coefficients an be eliminated in high compression/low quality images
- The spatial zero tree wavelet is used to achieve high compression rates with the wavelet transformed data

Self-Similarity Property





EZW Algorithm

- Compute the wavelet transform of the image
- Set a threshold T1 near the middle of the range of WT coefficient magnitudes. This gives a large "dead zone" that creates of lots of "insignificant values"
- These give rise to lots of zerotrees which efficiently handle significance map problem
- Send MSB's of significant coefficients. Use symbols P,N,Z, and t to represent positive, negative, zero, and zerotrees respectively
- Then reduce threshold: $Tj+1 = Tj/2$
- This causes some former insignificant coeff to become significant
- Repeat above steps, refine old significant by sending next finer bit

Questions?