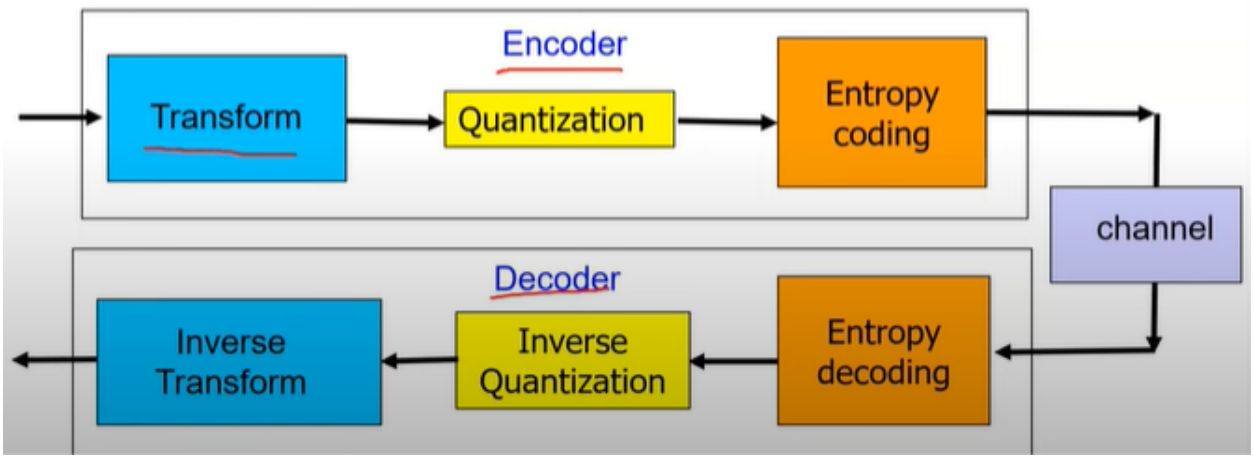
Lossy compression - March 23rd, 2024

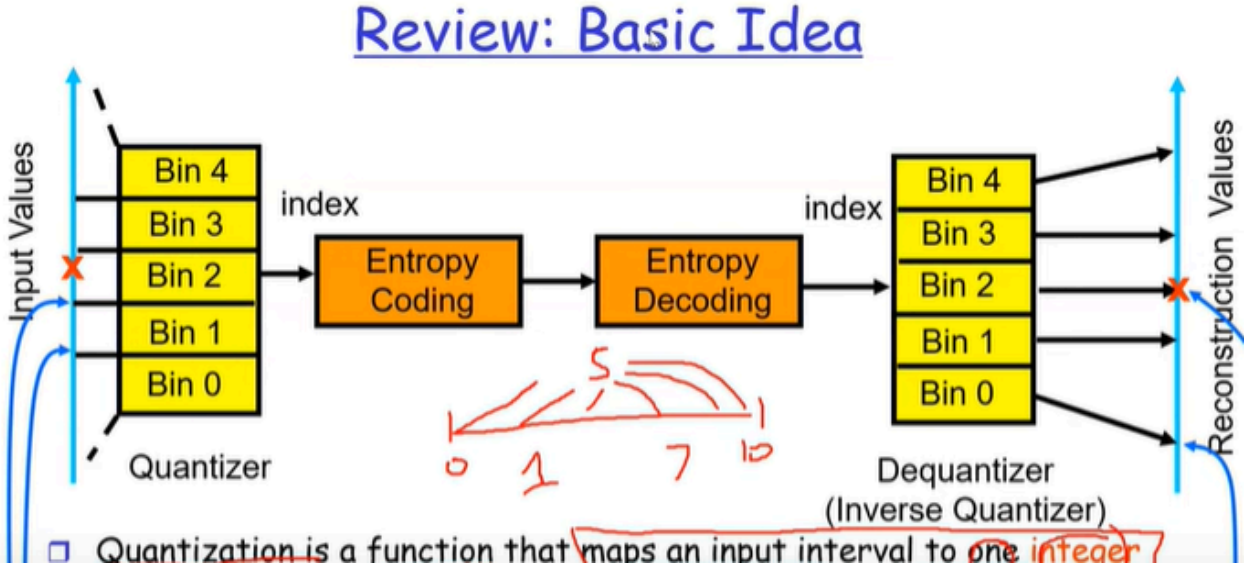
We need to find the middle ground between information lost and image quality.

Quantization

The process of representing a large (possibly infinite) set of values with a much smaller set.
A/D conversion is an efficient tool for lossy compression

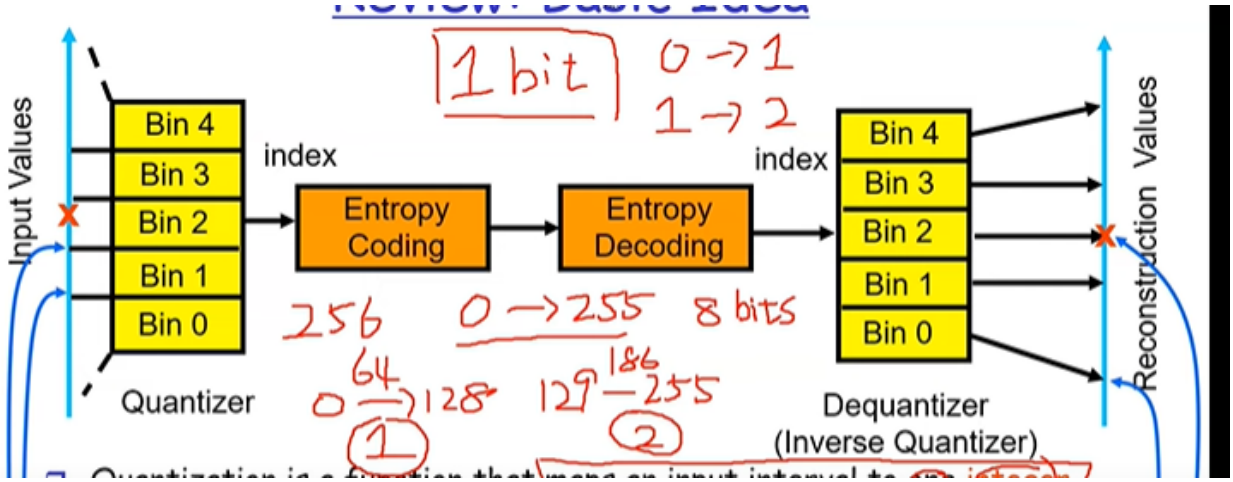


Transform is transform from spatial to frequency domain
Entropy coding is the lossless compression we talked previously about.



Reconstructed result is generally not the original input

Can reduce the bits required to represent the source.



In the inverse quanization, the values represent the quantized index, not the actual value.

Terminologies:

Decision boundaries bi: bin boundaries

Reconstruction levels yi: output value of each bin by the dequantizer.

Uniform Quantizer

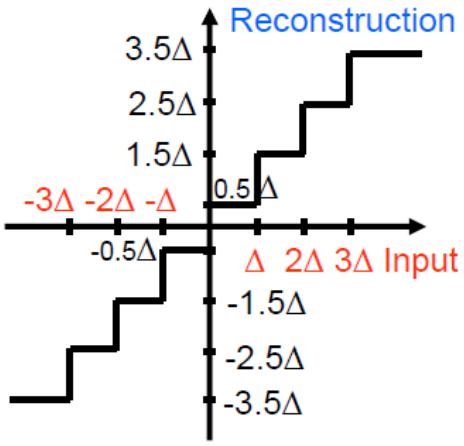
All bins have the same size except possibly for the two outer intervals:

bi and yi are spaced evenly

The spacing of bi and yi are both Δ (step size)

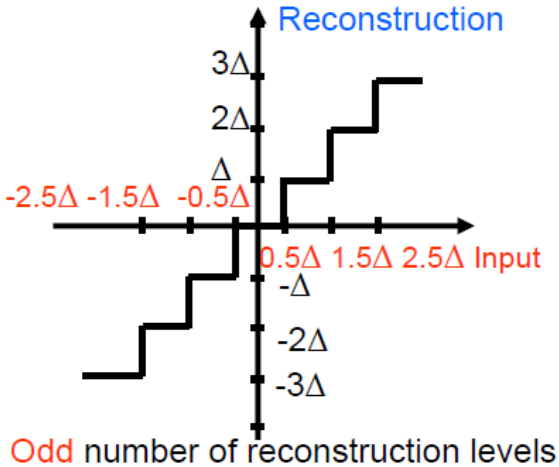
$$y_i = \frac{1}{2}(b_{i-1} + b_i) \quad \text{for inner intervals.}$$

Uniform Midrise Quantizer



Even number of reconstruction levels
0 is not a reconstruction level

Uniform Midtread Quantizer



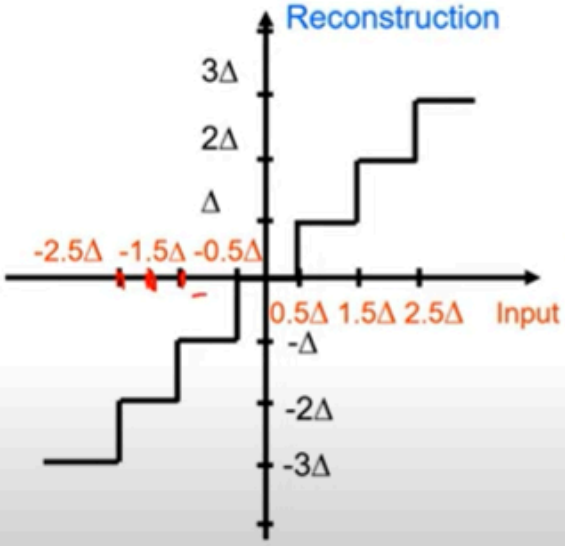
Odd number of reconstruction levels
0 is a reconstruction level

Midrise quantizer has equal number of bins for the positive and negative intervals.

Midtread Quantizer

Sign(x) { 1, x≥0 ; -1, x<0

- Quantization mapping:
  Output is an index



$$q = A(x) = sign(x)\left\lfloor \frac{|x|}{\Delta} + 0.5 \right\rfloor$$

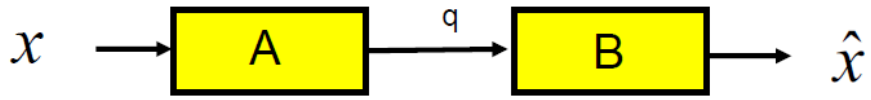
- Example:
  x = -1.8Δ, q = -2.

- De-quantization mapping:

$$\hat{x} = B(q) = q\Delta$$

-2Δ

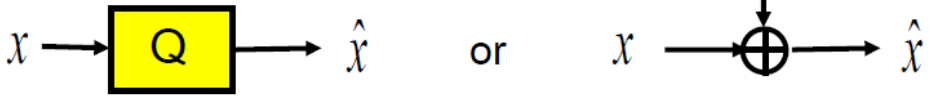
Model of Quantization



- Quantization: $q = A(x)$
- Inverse Quantization: $\hat{x} = B(q) = B(A(x)) = Q(x)$

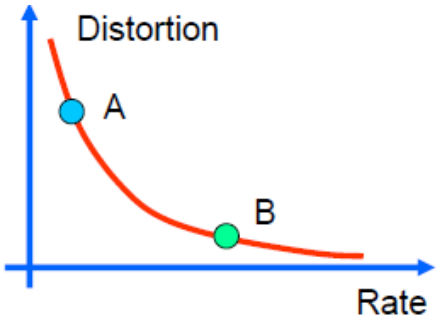
B(x) is not exactly the inverse function of A(x), because $\hat{x} \neq x$

- Quantization error: $e(x) = x - \hat{x}$
- Combining quantizer and de-quantizer:



When we are doing quanization, we need to decide some parameters. We need to decide: number of bins, bin boundaries and reconstruction levels.
Reconstruction levels - For uniform quantizer, reconstruction is the midpoint of the boundary.

- Things to be determined:
  - Number of bins
  - Bin boundaries
  - Reconstruction levels



- A tradeoff between rate and distortion:
  - To reduce the size of the encoded bits, we need to reduce the number of bins
  - Less bins ➔ More reconstruction errors

How do we measure distortion?

- Quantization error: $e(x) = x - \hat{x}$
- Mean Squared Error (MSE) for Quantization
  - Average quantization error of all input values
  - Need to know the probability distribution of the input

- Number of bins: M
- Decision boundaries: b_i, i = 0, …, M
- Reconstruction Levels: y_i, i = 1, …, M
- Reconstruction: $\hat{x} = y_i \quad \text{iff } b_{i-1} < x \le b_i$
- MSE: $$MSE_q = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2 f(x)dx$$
  - Same as the variance of e(x) if μ = E{e(x)} = 0 (zero mean).
  - Definition of Variance: $$\sigma_e^2 = \int_{-\infty}^{\infty} (e-\mu_e)^2 f(e)de$$

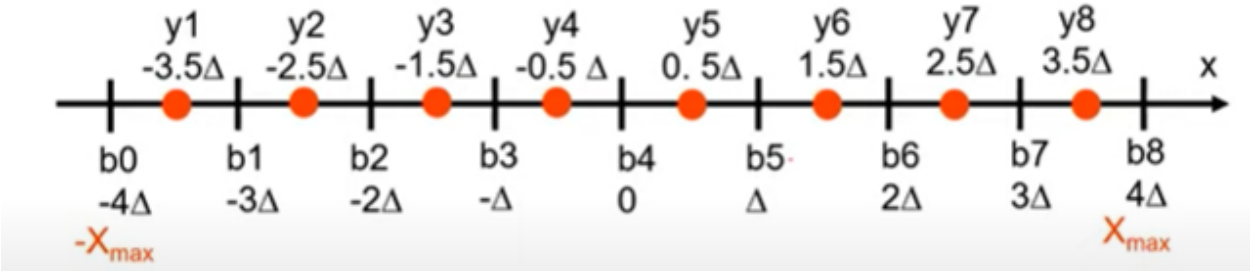
MSE - Mean Square Error
Two Scenarios:

1) Given M, find bi and yi that minimize the MSE.
2) Given a distortion constraint D, find M, bi and yi such that the MSE ≤ D.

Uniform vs. Non-Uniform Quantization

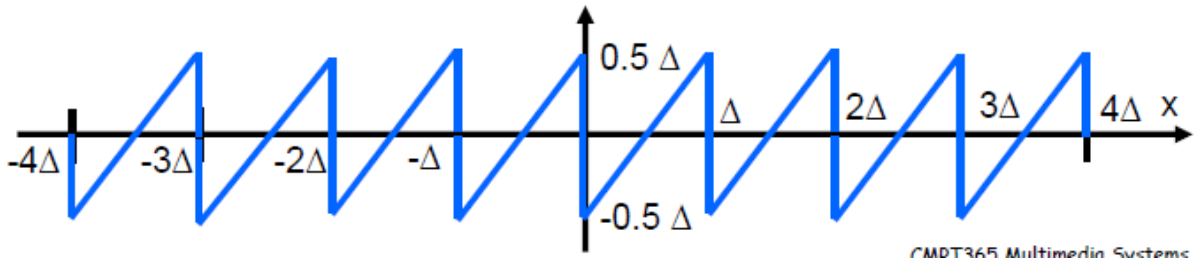
Uniform

- Input X: uniformly distributed in $[-X_{max}, X_{max}]$: $f(x) = 1 / (2X_{max})$
- Number of bins: M (even for midrise quantizer)
- Step size is easy to get: $\Delta = 2X_{max} / M$.
- $b_i = (i - M/2)\,\Delta$



f(x) means the probability for X to appear

- ➔ e(x) is uniformly distributed in $[-\Delta/2, \Delta/2]$.



How do we calculate the distortion for uniform quanization? MSE
Without the square in the MSE defn, we would get zero and the error's positive and negative components would cancel each other out

- MSE

$$MSE_q = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2 f(x)dx$$

$$= M \frac{1}{2X_{\max}} \int_0^{\Delta} \left(x - \frac{\Delta}{2}\right)^2 dx = \frac{M}{2X_{\max}} \frac{1}{12}\Delta^3 = \frac{1}{12}\Delta^2$$

  - M increases, Δ decreases, MSE decreases

- Variance of a random variable uniformly distributed in [- Δ/2, Δ/2]:

$$\sigma^2{}_q = \int_{-\Delta/2}^{\Delta/2} (x-0)^2 \frac{1}{\Delta} dx = \frac{1}{12}\Delta^2$$

- Optimization: Find M such that MSE ≤ D

$$\frac{1}{12}\Delta^2 \le D \Rightarrow \frac{1}{12}\left(\frac{2X_{\max}}{M}\right)^2 \le D \Rightarrow M \ge X_{\max}\sqrt{\frac{1}{3D}}$$

Come back to Signal to Noise Ratio (SNR) to measure quality of uniform quantization.

- Variance is a measure of signal energy
- Let M = 2^n
- Each bin index is represented by n bits

$$SNR(dB) = 10\log_{10}\frac{Signal\ Energy}{Noise\ Energy} = 10\log_{10}\frac{1/12(2X_{\max})^2}{1/12\Delta^2}$$

$$= 10\log_{10}\frac{(2X_{\max})^2}{(2X_{\max}/M)^2} = 10\log_{10} M^2 = 10\log_{10} 2^{2n} = (20\log_{10} 2)n$$

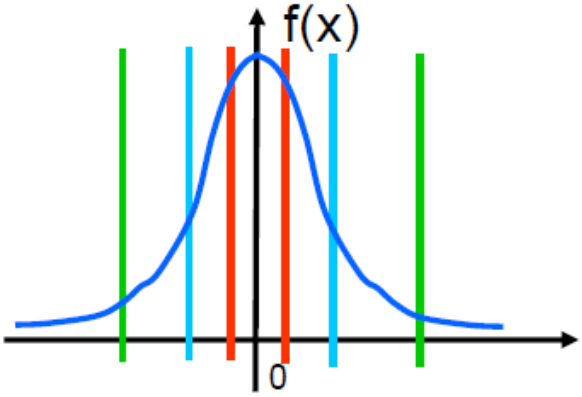
$$\approx 6.02n \ \ dB$$

- If n→n+1, Δ is halved, noise variance reduces to 1/4, and SNR increases by 6 dB.

Non-uniform Quantization

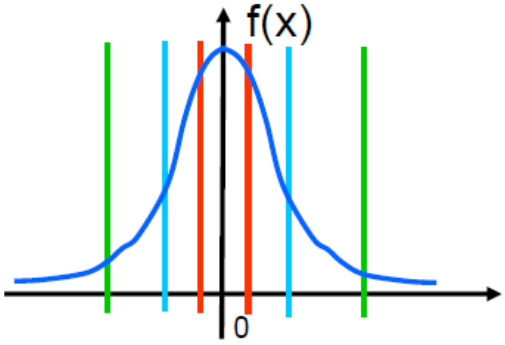
- Uniform quantizer is not optimal if source is not uniformly distributed
- For given M, to reduce MSE, we want narrow bin when f(x) is high and wide bin when f(x) is low

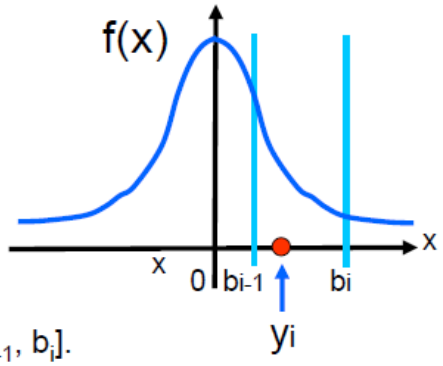
$$\sigma_q^2 = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x-y_k)^2 f(x)dx$$



Lloyd-Max Quantizer

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x-y_k)^2 f(x)dx$$





y_i is the centroid of interval $[b_{i-1}, b_i]$.

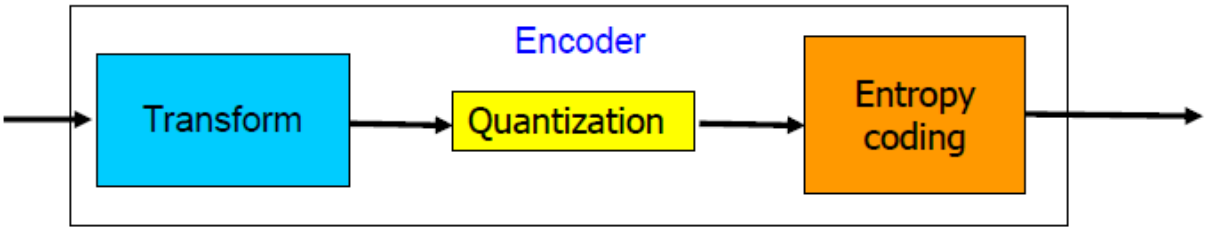
Discrete Cosine Transform (DCT)

Transform

- From one domain/space to another space
- Time -> Frequency
- Spatial/Pixel -> Frequency

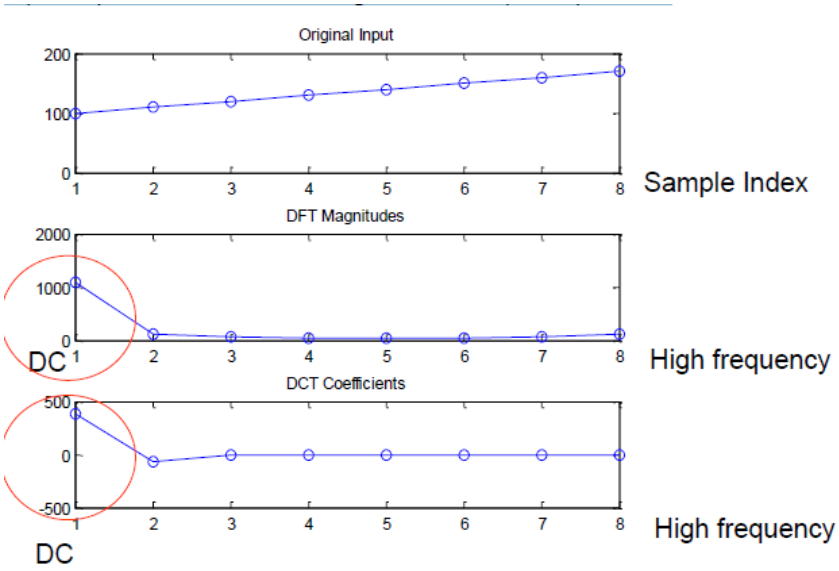
Purpose of transform

- Remove correlation between input samples
- Transform most energy of an input block into a few coefficients
- Small coefficients can be discarded by quantization without too much impact to reconstruction quality



The main idea behind transformation is that after doing the transform, we can remove the correlation behind the data. We can remove the components that have the small coefficients, having small coefficients means they don't have a big impact on the data.

Smooth signals have strong DC (direct current, or zero frequency) and low frequency components, and weak high frequency components.



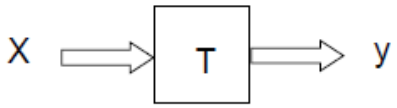
Looking at third figure (DCT), we can see the biggest coefficient is in the Direct Component, which stores the energy for the constant energy. As component # increases, we have higher frequency but near zero coefficients as

original input doesn't change much. We only really have to keep components 1 and 2.

Matrix Representation of Transform

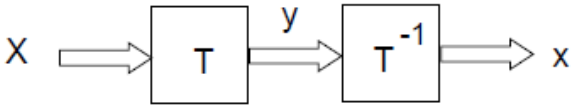
- Linear transform is an N x N matrix:

$$\mathbf{y}_{N\times 1} = \mathbf{T}_{N\times N}\mathbf{x}_{N\times 1}$$



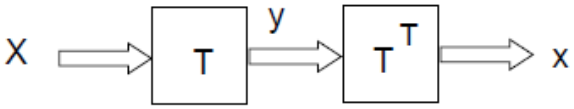
- Inverse Transform:

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{y}$$



- Unitary Transform (aka orthonormal):

$$\mathbf{T}^{-1} = \mathbf{T}^{T}$$



- For unitary transform: rows/cols have unit norm and are orthogonal to each others

$$\mathbf{T}\mathbf{T}^{T} = \mathbf{I} \Rightarrow \mathbf{t}_i\mathbf{t}_j^{T} = \delta_{ij} = \begin{cases} 1, & \mathrm{i} = \mathrm{j} \\ 0, & \mathrm{i} \neq \mathrm{j} \end{cases}$$

Now, how do we find transform matrix for DCT? DCT – close to optimal (known as KL Transform(orthogonal linear transform)) but much simpler and faster.

- Definition:

$$\mathbf{c}_{i,j} = a\cos\left(\frac{(2j+1)\,i\,\pi}{2N}\right), \quad \mathrm{i, j} = 0, \ldots, \mathrm{N}-1.$$

$$a = \sqrt{1/N} \quad \text{for i} = 0,$$

$$a = \sqrt{2/N} \quad \text{for i} = 1, \ldots, \mathrm{N}-1.$$

Matlab function: dct(eye(N))

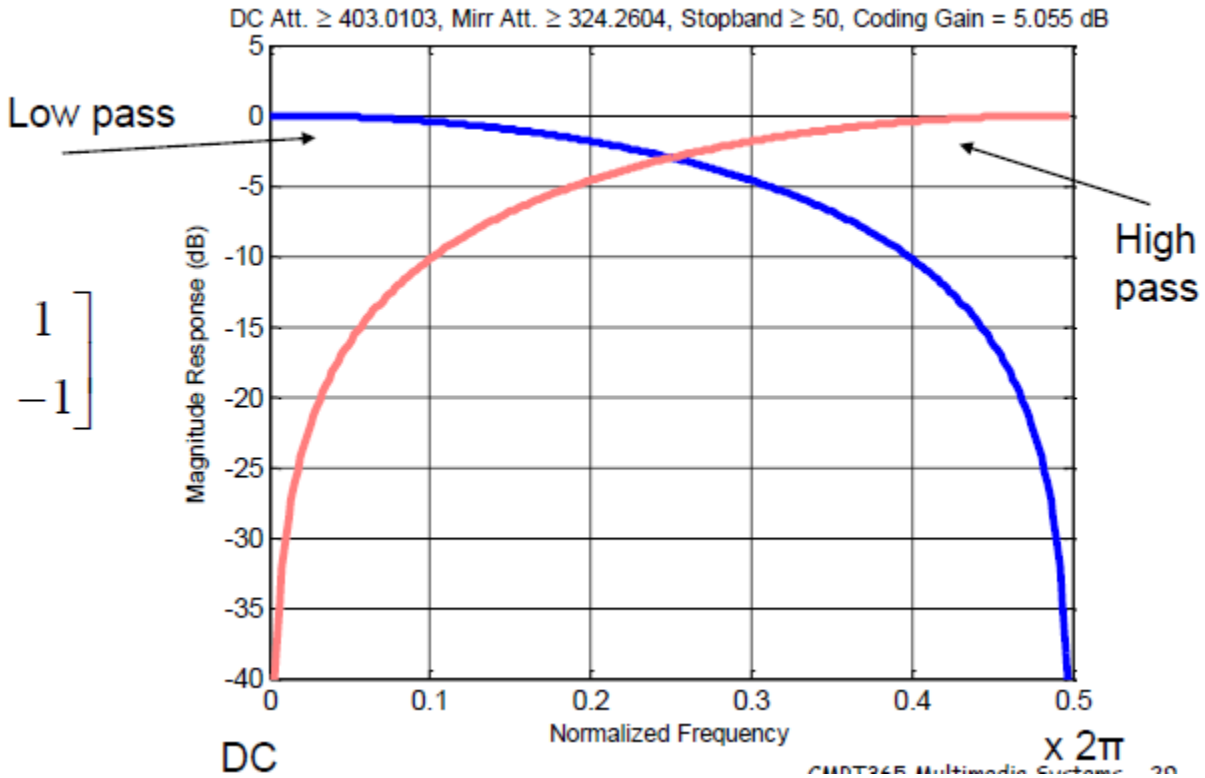
- N = 2 (Haar Transform):

$$\mathbf{C}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \mathbf{C}_2 \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} x_0 + x_1 \\ x_0 - x_1 \end{bmatrix}$$

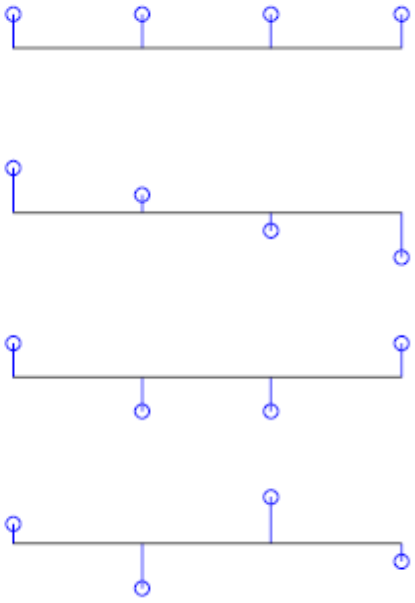
- y_0 captures the mean of x_0 and x_1 (low-pass)
  - $x_0 = x_1 = 1 \rightarrow y_0 = \text{sqrt}(2)$ (DC), $y_1 = 0$
- y1 captures the difference of x0 and x1 (high-pass)
  - $x_0 = 1,\ x_1 = -1 \rightarrow y_0 = 0$ (DC), $y_1 = \text{sqrt}(2)$.

Mean is not average, frequency filter have low pass and high pass filter.



We can see that for low frequencies (x axis) low pass filter will not filter the frequency but will filter the high frequency, and vice versa.

0.5000	0.5000	0.5000	0.5000
0.6533	0.2706	-0.2706	-0.6533
0.5000	-0.5000	-0.5000	0.5000
0.2706	-0.6533	0.6533	-0.2706

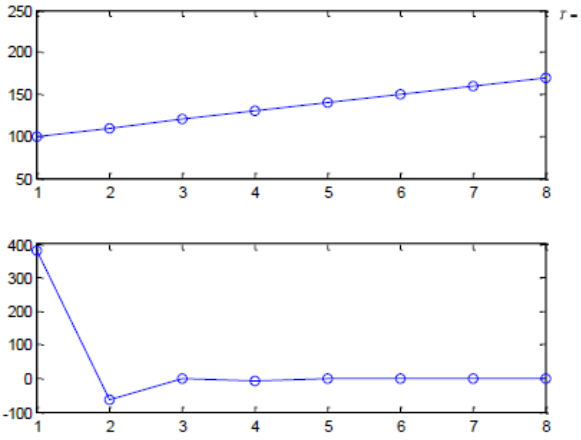


$$C = \sqrt{\frac{2}{(4)}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{2(4)} & \cos\frac{3\pi}{2(4)} & \cos\frac{5\pi}{2(4)} & \cos\frac{7\pi}{2(4)} \\ \cos\frac{2\pi}{2(4)} & \cos\frac{6\pi}{2(4)} & \cos\frac{10\pi}{2(4)} & \cos\frac{14\pi}{2(4)} \\ \cos\frac{3\pi}{2(4)} & \cos\frac{9\pi}{2(4)} & \cos\frac{15\pi}{2(4)} & \cos\frac{21\pi}{2(4)} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}}\cos\frac{\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{3\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{5\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{7\pi}{8} \\ \frac{1}{\sqrt{2}}\cos\frac{\pi}{4} & \frac{1}{\sqrt{2}}\cos\frac{3\pi}{4} & \frac{1}{\sqrt{2}}\cos\frac{5\pi}{4} & \frac{1}{\sqrt{2}}\cos\frac{7\pi}{4} \\ \frac{1}{\sqrt{2}}\cos\frac{3\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{9\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{15\pi}{8} & \frac{1}{\sqrt{2}}\cos\frac{21\pi}{8} \end{bmatrix}$$

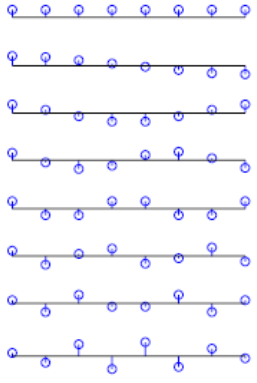
Each row of the transform matrix represents a certain frequency

- x = [100 110 120 130 140 150 160 170]^T;
- 8-point DCT:
  [381.8377, -64.4232, 0.0, -6.7345, 0.0, -2.0090, 0.0, -0.5070]
  Most energy are in the first 2 coefficients.



$T =$

.3536	.3536	.3536	.3536	.3536	.3536	.3536	.3536
.4904	.4157	.2778	.0975	−.0975	−.2778	−.4157	−.4904
.4619	.1913	−.1913	−.4619	−.4619	−.1913	.1913	.4619
.4157	−.0975	−.4904	−.2778	.2778	.4904	.0975	−.4157
.3536	−.3536	−.3536	.3536	.3536	−.3536	−.3536	.3536
.2778	−.4904	.0975	.4157	−.4157	−.0975	.4904	−.2778
.1913	−.4619	.4619	−.1913	−.1913	.4619	−.4619	.1913
.0975	−.2778	.4157	−.4904	.4904	−.4157	.2778	−.0975



Forward transform y = Tx (x is N by 1 vector)

Let ti be the i-th row of T
➔ yi = ti x = <ti^T, x> (Inner product)
yi measures the similarity between x and ti
Higher similarity ➔ larger transform coefficient

- Inverse transform:

$$\mathbf{x} = \mathbf{T}^T\mathbf{y} = \begin{bmatrix}\mathbf{t}_0^T & \mathbf{t}_1^T & \cdots & \mathbf{t}_{N-1}^T\end{bmatrix}\mathbf{y} = \sum_{i=0}^{N-1}\mathbf{t}_i^T y_i$$

- x is the weighted combination of t_i.
  - Rows of T are called basis vectors.

So far we have been transforming 1D data, what about 2D data?

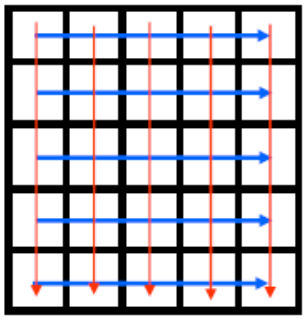
Apply the transform matrix to each row of the data (1D data) and then apply the transform again on each column.

The basis for 2D DCT ranges from low frequency to high frequency
Most transform coefficients are around 0.
Desired for compression

From 1D to 2D: Separable Transform

- X: N x N input block
- T: N x N 1D transform
- A = TX: Apply T to each column of X
- B=XT^T: Apply T to each row of X
- 2-D Separable Transform:
  - Apply T to each row
  - Then apply T to each column

$$\mathbf{Y} = \mathbf{T}\mathbf{X}\mathbf{T}^T$$



- Inverse Transform:

$$\mathbf{X} = \mathbf{T}^T\mathbf{Y}\mathbf{T}$$

- Not all 2D transforms are separable, but DCT is !

March 31st - Audio Compression

Audio uncompressed is too large.
We can do lossless

- Algorithm
  - Linear prediction: estimate what the value a sample will have, based on previous samples

$$x'(n) = \sum_{i=1}^{p} a_i x(n-i)$$

  - Channel coupling - mid/side-coding:
    - calculates a "mid"-channel by addition of left and right channel (l+r)/2 and a "side"-channel (l-r)/2.
  - Entropy coding

In linear prediction equation: x'(n) - prediction for n'th sample. A_i weight for previous sample. P is # of previous samples.

FLAC (Free Lossless Audio Codec)

- Golomb-Rice coding for entropy coding
  - closely related to Huffman/Arithmetic

Monkey's Audio – APE format

- Range coding for entropy coding
  - Similar to Arithmetic Coding
- Slightly better than FLAC (but time-consuming decoding)
- Officially only for Windows

Lossy coding: Perceptual Coding

- Hide errors where humans will not see or hear it
  - Study hearing and vision system to understand how we see/hear
  - Masking refers to one signal overwhelming/hiding another (e.g., loud siren or bright flash)
- Natural Bandlimitng
  - Audio perception is 20-20 kHz but most sounds in low frequencies (e.g., 2 kHz to 4 kHz)
  - Low frequencies may be encoded as single channel
  - Human ear can tolerate 200ps delay

Basically: If you can't hear the sound, don't encode it

- Frequency range is about 20 Hz to 20 kHz, most sensitive at 2 to 4 KHz.
- Dynamic range (quietest to loudest) is about 96 dB
    - 16 bits is good enough for quantization
- Normal voice range is about 500 Hz to 2 kHz
  • Low frequencies are vowels and bass
  • High frequencies are consonants

There are several properties of the Psychoacoustic Model:

- Temporal masking: If we hear a loud sound, it takes a little while until we can hear a soft tone nearby.
- Frequency masking: If within a critical band a stronger sound and weaker sound compete, you can't hear the weaker sound. Don't encode it

Perceptual coding makes use of psychoacoustic knowledge to reduce the amount of information required to achieve the same perceived quality (lossy compression). So like MPEG (MP3).

MP3 (MPEG)

MPEG-1: 1.5 Mbits/sec for audio and video

- About 1.2 Mbits/sec for video, 0.3 Mbits/sec for audio
- Uncompressed CD audio is 44,100 samples/sec * 16 bits/sample * 2 channels > 1.4 Mbits/sec
- Compression factor ranging from 2.7 to 24.
- With Compression rate 6:1 (16 bits stereo sampled at 48 KHz is reduced to 256 kbits/sec), expert could not distinguish

Layer 1: DCT type filter with one frame and equal frequency spread per band. Psychoacoustic model only uses frequency masking.
Layer 2: Three frames in filter (before, current, next, a total of 1152 samples). This models a little bit of the temporal masking.

Layer 3 (MP3): Better critical band filter is used (non-equal frequencies), psychoacoustic model includes temporal masking effects (frame before and after), takes into account stereo redundancy, and uses Huffman coder.

Stereo Redundancy Coding:

Intensity stereo coding -- at upper-frequency subbands, encode summed signals instead of independent signals from left and right channels.
Middle/Side (MS) stereo coding -- encode middle (sum of left and right) and side (difference of left and right) channels.

Algorithm

Divide the audio signal (e.g., 48 kHz sound) into 32 frequency subbands --> subband filtering.

Modified discrete cosine transform (MDCT)

Masking for each band caused by nearby band

- psychoacoustic model
- If the power in a band is below the masking threshold, don't encode it.
- Otherwise, determine number of bits needed to represent the coefficient such that noise introduced by quantization is below the masking effect
  • One fewer bit introduces about 6 dB of noise).

❒ After analysis, the first levels of 16 of the 32 bands:

Band	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Level (db)	0	8	12	10	6	2	10	60	35	20	15	2	3	5	3	1

If the level of the 8th band is 60dB, it gives a masking of 12 dB in the 7th band, 15dB in the 9th.
Level in 7th band is 10 dB (< 12 dB), so ignore it.
Level in 9th band is 35 dB (> 15 dB), so send it. We don’t need to send 35 dB, we can just send 35 - 15 = 20 dB. Meaning we use SQNR equation 6.02*N, N number of bits. 6.02*4 = 24 dB so we can use 4 bits.

Layer	Target Bit-rate	Ratio	Quality at 64 kb/s	Quality at 128 kb/s	Theoretical Min. Delay
Layer 1	192 kb/s	4:1	---	---	19 ms
Layer 2	128 kb/s	6:1	2.1 to 2.6	4+	35 ms
Layer 3	64 kb/s	12:1	3.6 to 3.8	4+	59 ms

Compression for image

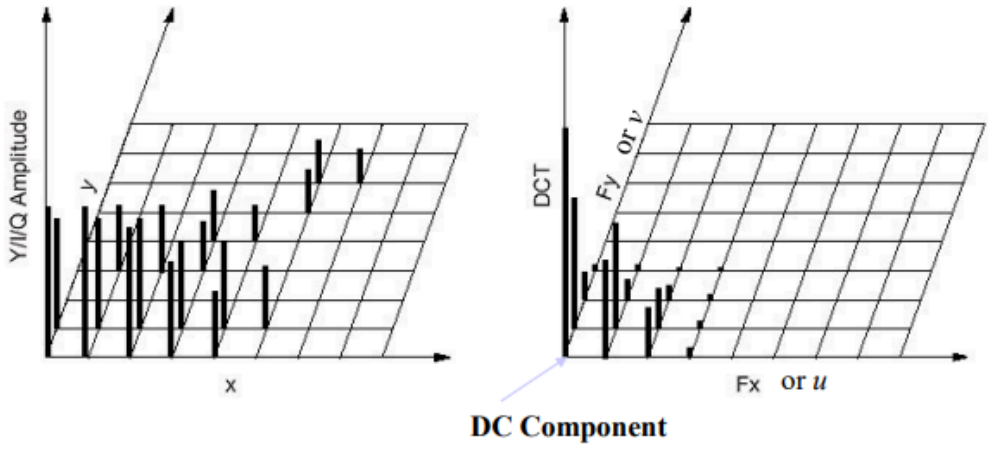
Observation 1: Useful image contents change relatively slowly across the image (same color value for a vast part of the image)

Observation 2: Psychophysical experiments suggest that humans are much less likely to notice the loss of very high spatial frequency components than the loss of lower frequency components.

- the spatial redundancy can be reduced by largely reducing the high spatial frequency contents.

High frequency components: edges or color changes

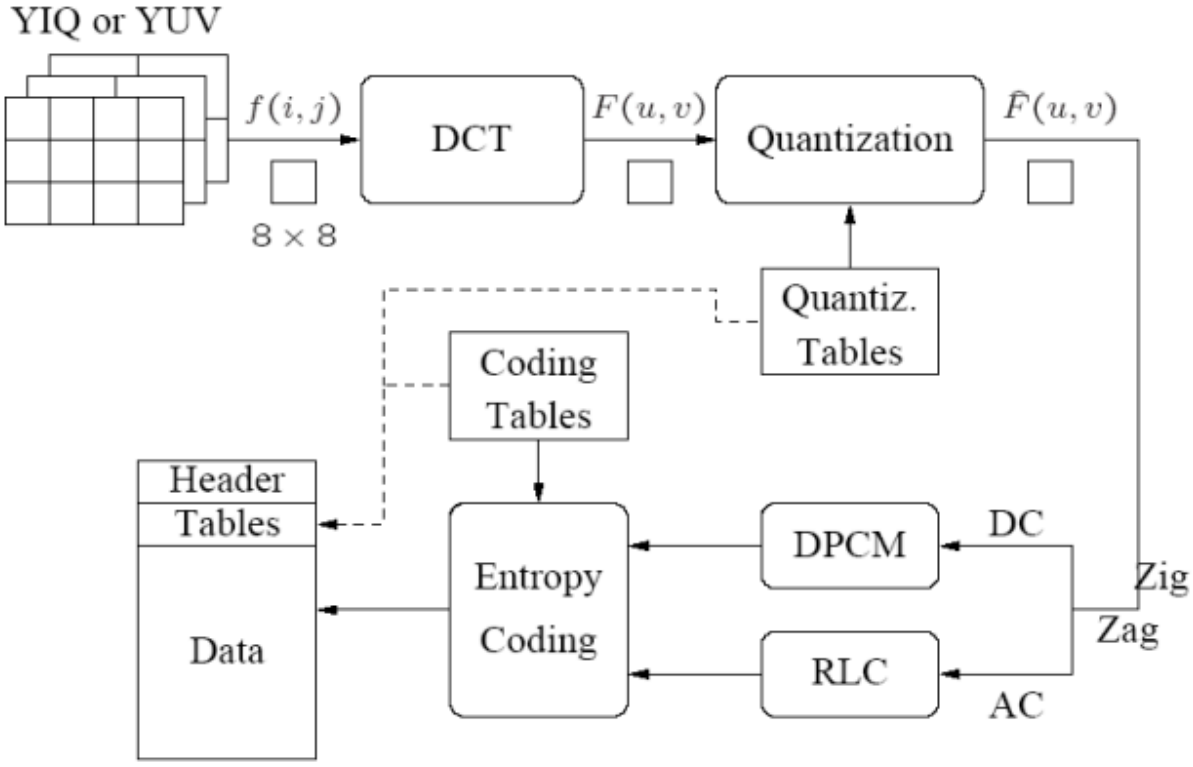
Observation 3: More sensitive for grayscale rather than color. We can distinguish closely spaced lines in grayscale better than color.



Original values of an 8x8 block (in spatial domain)

Corresponding DCT coefficients (in frequency domain)

1) Block Preparation
   - RGB to YUV (YIQ) planes
2) Transform
   - 2D Discrete Cosine Transform (DCT) on 8x8 blocks.
3) Quantization
   - Quantized DCT Coefficients (lossy).
4) Encoding of Quantized Coefficients
   a) Zigzag Scan
   b) Differential Pulse Code Modulation (DPCM) on DC component
   c) Run Length Encoding (RLE) on AC Components
   d) Entropy Coding: Huffman or Arithmetic



1) Input image: 640 x 480 RGB (24 bits/pixel) transformed to three planes: Y: (640 x 480, 8-bit/pixel) Luminance (brightness) plane. U, V: (320 X 240 8-bits/pixel) Chrominance (color) planes. These are all divided into 8x8 blocks.
   a) The higher the compression ratio is, the blocky the final image (and badly compressed) the final image will be
2) 3

3) Uniform quantization: Divide by constant N and round result. In JPEG, each DCT F[u,v] is divided by a constant q(u,v). The table of q(u,v) is called quantization table

Quantization table

q(u,v)

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

DCT Coefficients

F[u,v]

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	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

Rounded F[u,v]/ Q(u,v)

Quantization is the main source for loss

- Q(u, v) of larger values towards lower right corner
- More loss at the higher spatial frequencies
  • Supported by Observations 1 and 2.
- Q(u,v) obtained from psychophysical studies
  • maximizing the compression ratio while minimizing perceptual losses

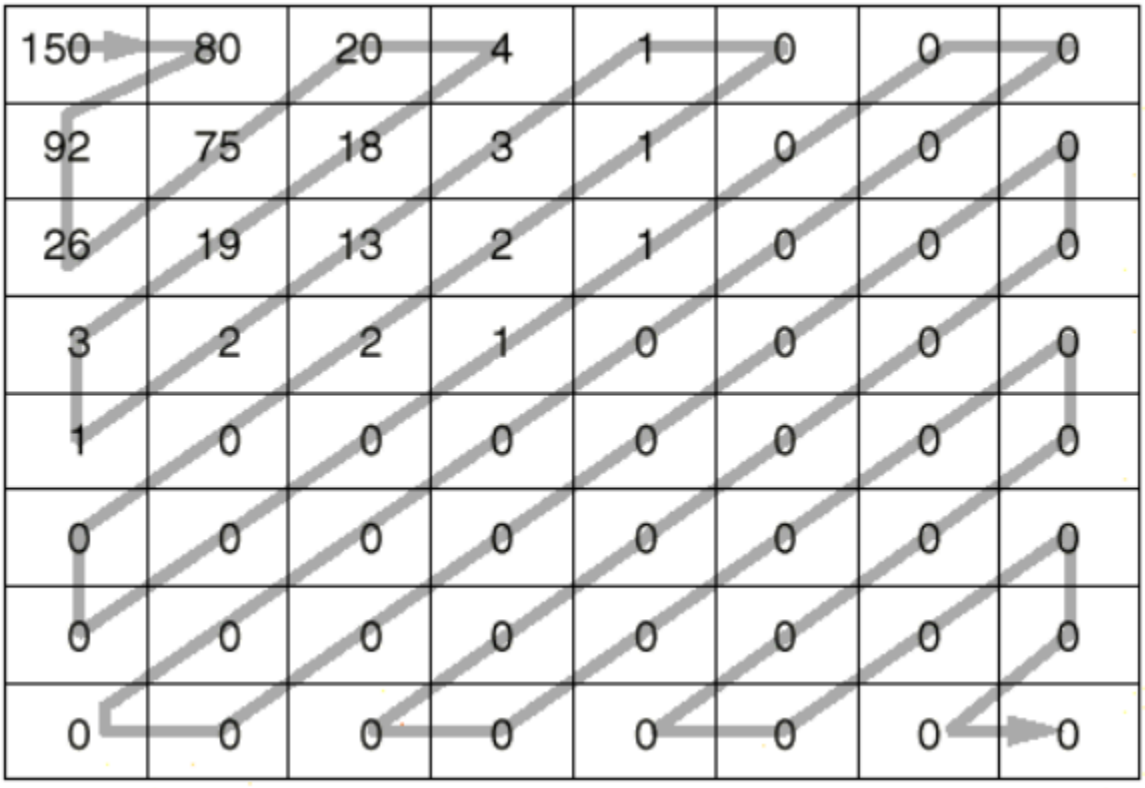
4) Encoding: We have two components.

- DC Components (zero frequency)
  - DC component of a block is large and varied, but often close to the DC value of the previous block.

- Encode the difference from previous
  • Differential Pulse Code Modulation (DPCM).
- AC components: Lots of zeros (or close to zero)
  - Run Length Encoding (RLE, or RLC).
    • encode as (skip, value) pairs
    • Skip: number of zeros, value: next non-zero component (0,0) as end-of-block value.
  - Example 000030002: (4,3), (3,2), (0,0)

Zigzag Scan

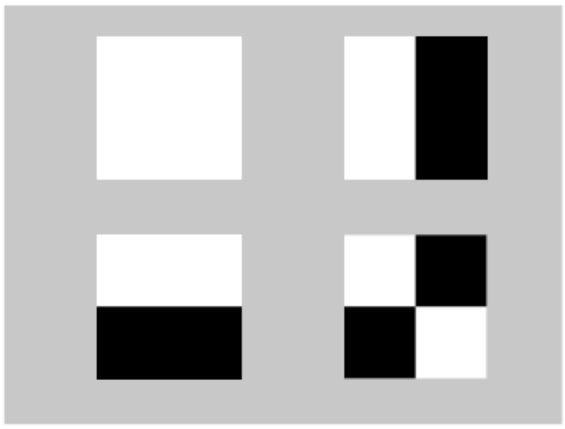
A method to scan the DCT coefficient.
Maps an 8x8 block into a 1 x 64 vector Zigzag pattern group low frequency coefficients in top of vector.



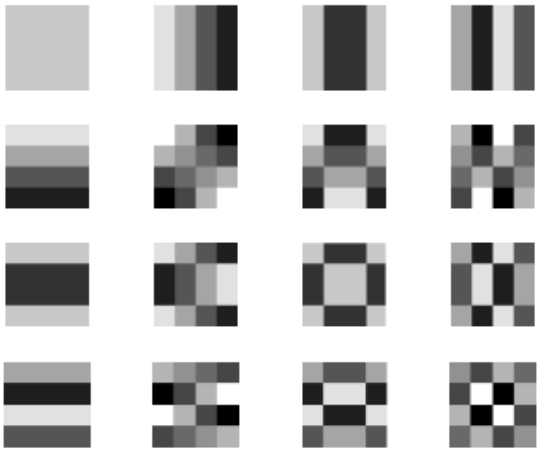
Reading by rows will won't give us a lot of consecutive zeroes. Doing zig zag scan will give more consecutive zeroes so run length coding will be more efficient.

Recall: 2-D DCT Basis Matrices

For 2-point DCT



For 4-point DCT



Frequency increases to the bottom right.

A typical 8x8 block of quantized DCT coefficients.
Most of the higher order coefficients have been quantized to 0.

12	34	0	54	0	0	0	0
87	0	0	12	3	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

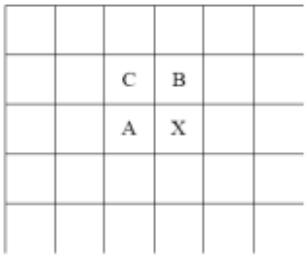
Zig-zag scan: the sequence of DCT coefficients to be transmitted:
12 34 87 16 0 0 54 0 0 0 0 0 0 12 0 0 3 0 0 0 .....
DC coefficient (12) is sent via a separate Huffman table.
Runlength coding remaining coefficients:
34 | 87 | 16 | 0 0 54 | 0 0 0 0 0 0 12 | 0 0 3 | 0 0 0 .....

JPEG also uses Huffman/arithmetic coding as the last lossless step. What about if we wanted to make the whole process lossless?

Lossless Version of JPEG

- A special mode; different from the standard JPEG
  - Using prediction and entropy coding only
- Forming a differential prediction:
  - A predictor combines the values of up to three neighboring pixels as the predicted value for the current pixel
  - 7 schemes for combination
- Encoding:
  - The encoder compares the prediction with the actual pixel value at the position `X' and encodes the difference using entropy coding

Predictor	Prediction
P1	A
P2	B
P3	C
P4	A + B - C
P5	A + (B - C) / 2
P6	B + (A - C) / 2
P7	(A + B) / 2



110 112
103 109

i.e., C=110, B=112, A=103, X=109

P1 6
P2 -3
P3 -1
P4 4
P5 5
P6 0.5
P7 1.5

Here, we can use predictor 1 to 7 and we are trying to predict X.

April 13th - Video compression

How to compress?

- Spatial redundancy - compression on each individual mage
- Temporal redundancy - prediction based on previous images

Frame 1

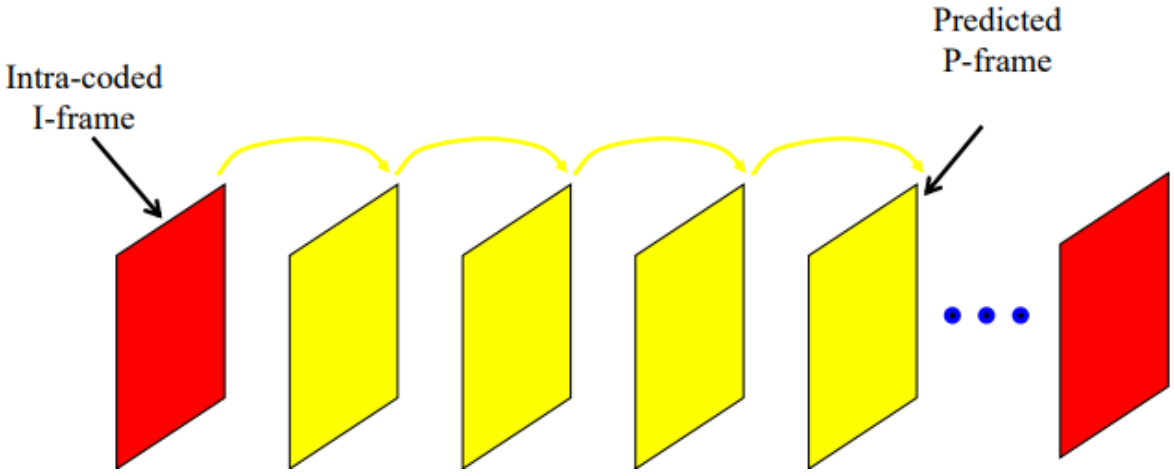
Frame 2

Direct Difference

Characteristics of typical videos:
A lot of similarities between adjacent frames
Differences caused by object or camera motion

Key idea: Predict each frame from the previous frame and only encode the prediction error: Pred. error has smaller energy and is easier to compress

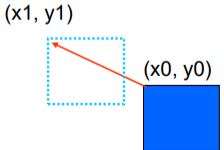
P-frames need I-frames to exist. I-frames can exist on their own.



We need several I-frames to prevent P-frames bottle neck and reduce latency. What are these prediction frames?

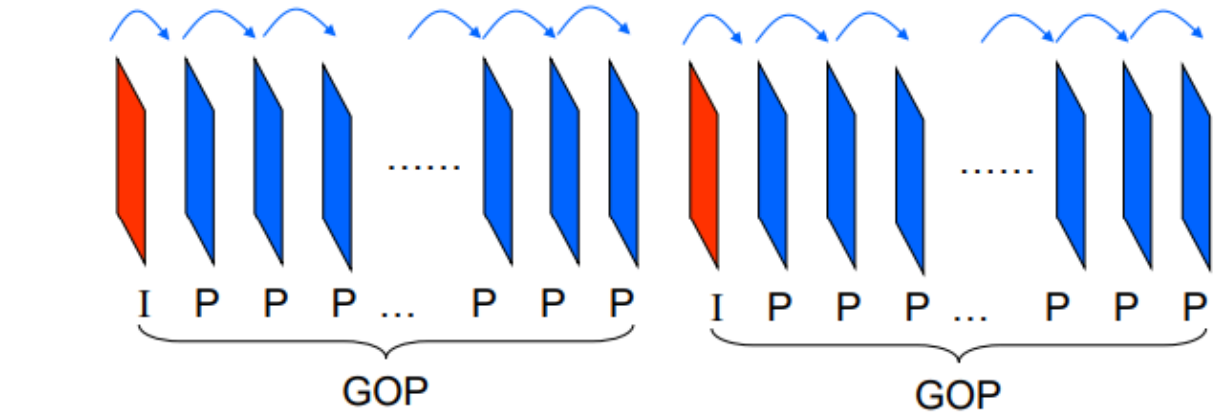
This is matching with a motion vector.

- For each block, find the best match in the previous frame (reference frame)
  - Upper-left corner of the block being encoded: (x0, y0)
  - Upper-left corner of the matched block in the reference frame: (x1, y1)
  - Motion vector (dx, dy): the offset of the two blocks:
    - (dx, dy) = (x1 – x0, y1 – y0)
    - (x0, y0) + (dx, dy) = (x1, y1)
  - Motion vector need to be sent to the decoder.



Given a reference frame and the motion vector, we can obtain a prediction of the current frame.
Prediction error: Difference between the current frame and the prediction. We can take the current frame - (reference frame + motion vector).
The prediction error will be coded by DCT, quantization, and entropy coding.

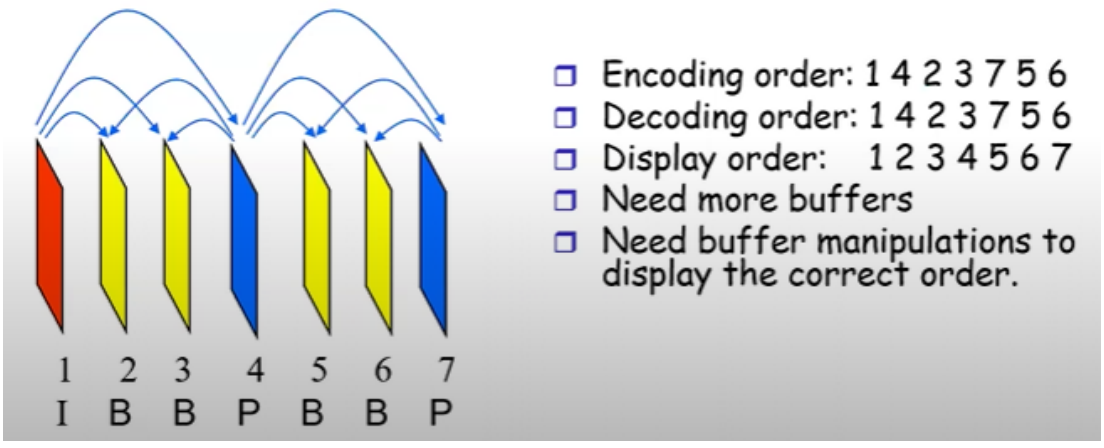
P frame will be encoded in standard image prediction.



- GOP: Group of pictures (frames).
- I frames (Key frames):
  - Intra-coded frame, coded as a still image. Can be decoded directly.
  - Used for GOP head, or at scene changes.
  - I frames also improve the error resilience.
- P frames: (Inter-coded frames)
  - Predication-based coding, based on previous frames.

B frames: Bi-directional interpolated prediction frames.

Predicted from both the previous frame and the next frame:
more flexibilities ➔ better prediction.
B frames are not used as reference for future frames: B frames can be coded with lower quality or can be discarded without affecting future frames.



Frame 2 for

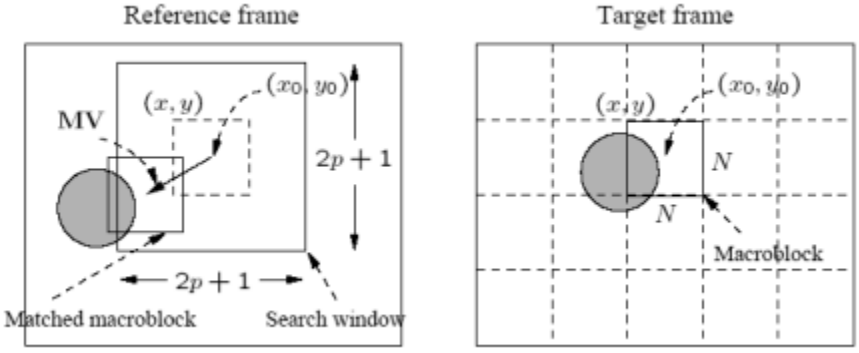
example, it will use I frame 1 and P frame 4 but not frame 3 as it's not used as a reference for future references.

Encoding order: I frame, next P frame, then B frames in between them, next P frame, then B frames between first and second P frame.

Decoding order: I frame, next P frame, in between B frames, next P frame, in between B frames.

Need more buffers as decoding frame 1 and 4, we are a frame behind in display order. If we have a lot of RAM, buffers are easy to implement. Not practical for memory limited scenarios.

Motion revisited: For an offset (i,j) in a search window (-p,p), calculate difference between original and moved macro blocks.



Mean square error (MSE) across all the pixels of the macroblock.

Mean absolute distance (MAD) across all the pixels of the macroblock.

$$MAD(i,j) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)|$$

N – size of the macroblock,

k and l – indices for pixels in the macroblock,

i and j – horizontal and vertical displacements,

$C(x+k, y+l)$ – pixels in macroblock in Target frame,

$R(x+i+k, y+j+l)$ – pixels in macroblock in Reference frame.

Full-search motion estimation is time consuming: Each (i, j) candidate: N^2 summations If search window size is W^2, need W^2 x N^2 comparisions / MB
• W=2p+1=31, N=16: ➔ 246016 comparisons / MB !

We need faster motion, like log search. Similar to binary search

Procedure – similar to a binary search
Initially, only nine locations in the search window are used as seeds for a MAD-based search; marked as `1'.
After the one that yields the minimum MAD is located, the center of the new search region is moved to it and the step-size (“offset") is reduced to half.
In the next iteration, the nine new locations are marked as `2', and so on.