ENEE408G Lecture-3

Image Compression

- URL: http://www.ece.umd.edu/class/enee408g/
- Slides included here are based on Spring 2012 offering in the order of introduction, image, video, speech, and audio. © Copyrighted 2002-2012.
- ENEE408G course was developed @ ECE Department, University of Maryland, College Park. Inquiries can be addressed to Profs. Ray Liu (kjrliu@umd.edu) and Min Wu (minwu@umd.edu).

Last Lecture

- Human visual properties
- Image enhancement
  - Visual quantization and dithering
  - Contrast stretching and histogram equalization
  - Noise removal via LPF filtering and median filtering
  - Image Sharpening
- Edge detection
- Today: image compression

  (follow-up) Image enlargement / interpolation

Why Need Compression?

- Savings in storage and transmission
  - Multimedia data (esp. image and video) have large data volume
  - Difficult to send real-time uncompressed video over current network
- Accommodate relatively slow storage devices
  - In case that they do not allow playing back uncompressed multimedia data in real time
    - *1x CD-ROM transfer rate ~ 150 kB/s*
    - *320 x 240 x 24 fps color video bit rate ~ 5.5MB/s*

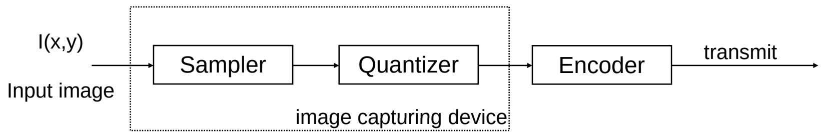
    => 36 seconds needed to transfer 1-sec uncompressed video from CD

List of Compression Tools

- Lossless encoding tools
  - Entropy coding: Huffman, Lemple-Ziv, and others
  - Run-length coding
- Lossy tools for reducing redundancy
  - Quantization and truncations
- Signal analysis/processing tools to help exploit redundancy
  - Predictive coding
    - *Encode prediction parameters and residues with less bits*
  - Transform coding
    - *Transform into a domain with improved energy compaction*

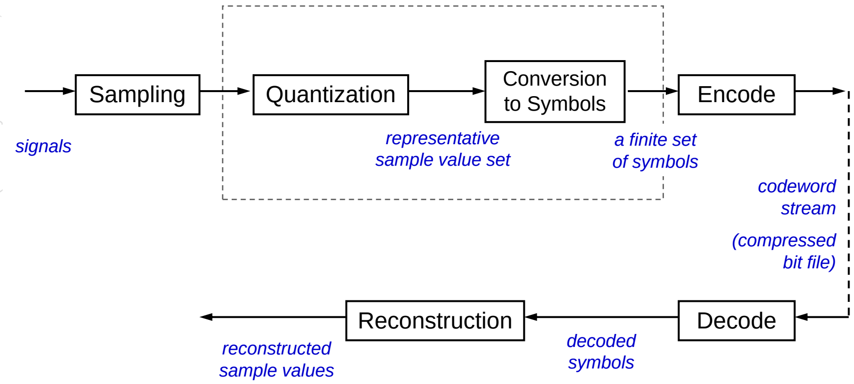
PCM coding

- How to encode a digital image into bits?
  - Sampling and perform uniform quantization
    - *"Pulse Coded Modulation" (PCM)*
    - *8 bits per pixel ~ good for grayscale image/video*
    - *10-12 bpp ~ needed for medical images*



- Reduce # of bpp for reasonable quality via quantization
  - Quantization reduces # of possible levels to encode
  - Visual quantization to reduce artifacts at low pixel depth:
    - *Dithering, companding, contrast quantization, etc.*
    - *Halftone use 1bpp but usually upsampling (trade spatial resolution with pixel depth) ~ savings less than 2:1*

CODEC System: enCOding and DECoding



Discussion on Improving PCM

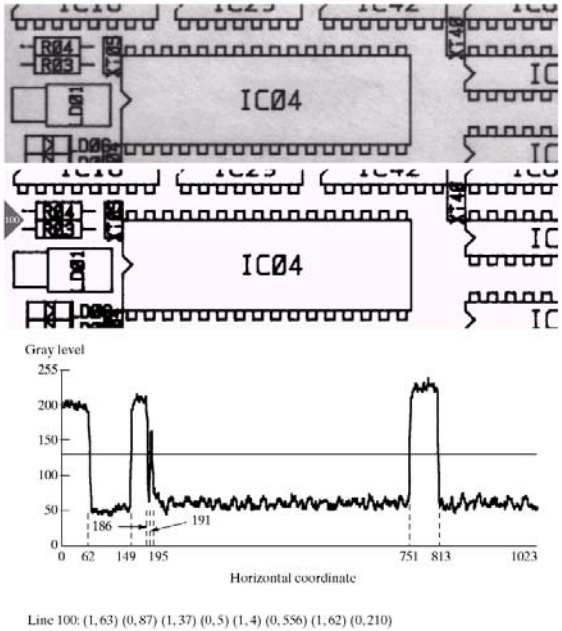
- Quantized PCM values may not be equally likely
  - Can we do better than encode each value using same # bits?
- Example
  - P("0") = 0.5, P("1") = 0.25, P("2") = 0.125, P("3") = 0.125
  - If use same # bits for all values

    => Need 2 bits to represent the four possibilities
  - If use less bits for likely values "0" ~ **Variable Length Codes (VLC)**
    - *"0" => [0], "1" => [10], "2" => [110], "3" => [111]*
    - *Use* $\boldsymbol{\Sigma_i\, p_i\, l_i}$ *= 1.75 bits on average (i.e the expected length) ~ saves 0.25 bpp!*
- Bring probability into the picture
  - Use prob. distr. to reduce average # bits per quantized sample

Entropy Coding

- Idea: use fewer bits for commonly seen values
    - *Challenge: prevent ambiguity and achieve efficiency in decoding*
- Examples:
  - Huffman coding (used in JPEG and MPEG)
    - *Build a codebook beforehand based on data statistics*
  - Lemple-Ziv coding (used in Unix)
    - *Collect statistics and build codebook in run-time*
- How many # bits needed?
  - "Compressability" depends on the source's characteristics
  - Limit of compression => "Entropy"
    - *Measures the uncertainty, or amount of avg. information of a source*

RLC Example



a b c d

FIGURE 8.3 Illustration of run-length coding: (a) original image. (b) Binary image with line 100 marked. (c) Line profile and binarization threshold. (d) Run-length code.

Figure is from slides at Gonzalez/ Woods DIP book website (Chapter 8)

Coding a Sequence of Bits

- How to efficiently encode it?
  - e.g. a row in a binary document image:
  - "0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 …"
- Run-length coding (RLC)
  - Code length of runs of "0" between successive "1"
    - *run-length of "0" ~ # of "0" between "1"*
    - *good if often getting frequent large runs of "0" and sparse "1"*
  - E.g. => (7) (0) (3) (1) (6) (0) (0) … …
  - Assign fixed-length codeword to run-length
  - Or use variable-length code like Huffman to further improve
- RLC can be used to non-binary data sequence with long run of "0"

a
b c

FIGURE 8.20 (a) The prediction error image resulting from Eq. (8.4-9). (b) Gray-level histogram of the original image. (c) Histogram of the prediction error.

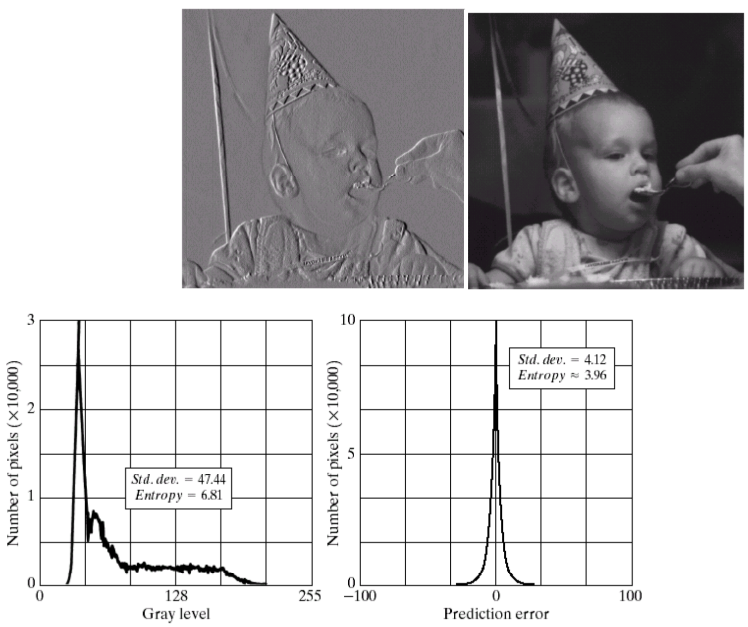
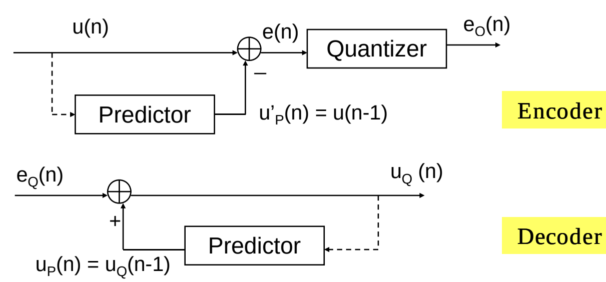


Figure is from slides at Gonzalez/ Woods DIP book website (Chapter 8). Use "previous pixel predictor". Difference image has mid-range gray representing zero and amplifying factor of 8.

How to Encode Correlated Sequence?

- Consider: high correlation between successive samples
- Predictive coding
  - Basic principle: remove redundancy between successive pixels and only encode residual between actual and predicted
  - Residue usually has much smaller dynamic range
    - *Allow fewer quantization levels for the same MSE => get compression*
  - Compression efficiency depends on intersample redundancy
- First try

Simple case: Differential PCM (DPCM) coding, i.e. use previous sample as our estimation of next sample.

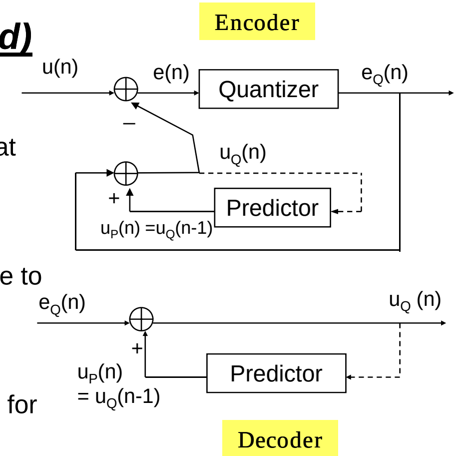


Drifting by "Open-Loop" Predictive Coder

Example: quantization step size Q = 5 with reconstruction points kQ

[Original signal]	98	101	99	97	95	…
Open-loop DPCM	98	+3	-2	-2	-2	…
Encoded (kQ)	20Q	+Q	0	0	0	…
Decoded	100	105	105	105	105	…
[Original signal]	98	101	99	97	95	…
Closed-loop DPCM	98	+1	-1	-3	0	…
Encoded (kQ)	20Q	0	0	-Q	0	…
Decoded	100	100	100	95	95	…

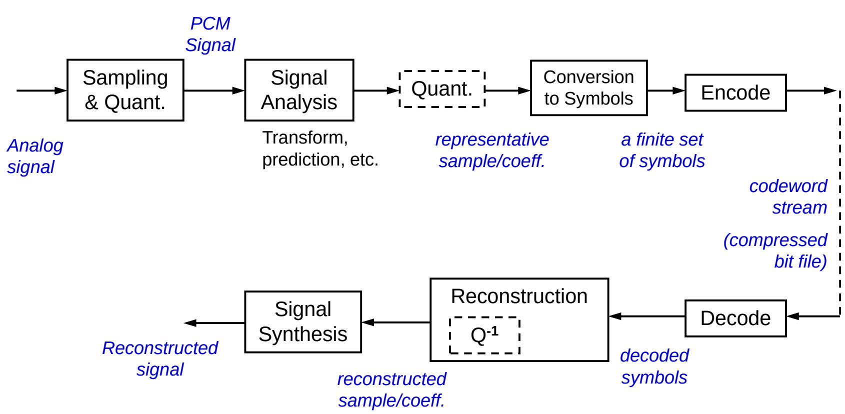
Predictive Coding (cont'd)



- Problem with 1st try
  - Input to predictor are different at encoder and decoder
    - *decoder doesn't know u(n)!*
  - Mismatch error could propagate to future reconstructed samples
- Solution:
  - Use quantized sequence $u_Q(n)$ for prediction at both encoder and decoder
  - Prediction error $e(n)$
  - Quantized prediction error $e_Q(n)$
  - Distortion $d(n) = e(n) - e_Q(n)$

Think: what predictor is good to use?

Revisit "CODEC" system: enCOding + DECoding



1-D Discrete Cosine Transform (DCT) @

$$\begin{cases} Z(k) = \sum_{n=0}^{N-1} z(n) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \\ z(n) = \sum_{k=0}^{N-1} Z(k) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \end{cases}$$

$$\alpha(0) = \frac{1}{\sqrt{N}}, \alpha(k) = \sqrt{\frac{2}{N}}$$

- DCT is a linear, orthogonal transform
- DCT is not the real part of DFT!
  - DCT is related to DFT of a symmetrically extended signal
- Represent a signal vector $\underline{z}$ using a set of orthonormal basis vectors $\{\underline{C}_k\}$
  - $\underline{z} = \Sigma Z(k) \underline{C}_k$
  - DCT coefficients {Z(k)} of a real-valued signal {z(n)} are real valued

Review and Examples of Basis

- Standard basis vectors

$$\begin{bmatrix}6\\3\\1\end{bmatrix} = 6\cdot\begin{bmatrix}1\\0\\0\end{bmatrix} + 3\cdot\begin{bmatrix}0\\1\\0\end{bmatrix} + 1\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}$$

- Standard basis images

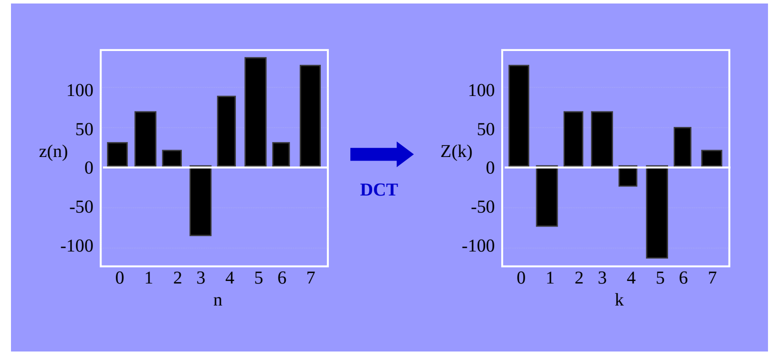
$$\begin{bmatrix}2 & 2\\3 & 0\end{bmatrix} = 2\cdot\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} + 2\cdot\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix} + 3\cdot\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + 0\cdot\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$$

- Example: representing a vector with different basis

$$\begin{bmatrix}3\\5\end{bmatrix} = 3\cdot\begin{bmatrix}1\\0\end{bmatrix} + 5\cdot\begin{bmatrix}0\\1\end{bmatrix} = 4\cdot\begin{bmatrix}1\\1\end{bmatrix} + 1\cdot\begin{bmatrix}-1\\1\end{bmatrix} = 4\sqrt{2}\begin{bmatrix}\sqrt{2}/2\\\sqrt{2}/2\end{bmatrix} + \sqrt{2}\begin{bmatrix}-\sqrt{2}/2\\\sqrt{2}/2\end{bmatrix}$$

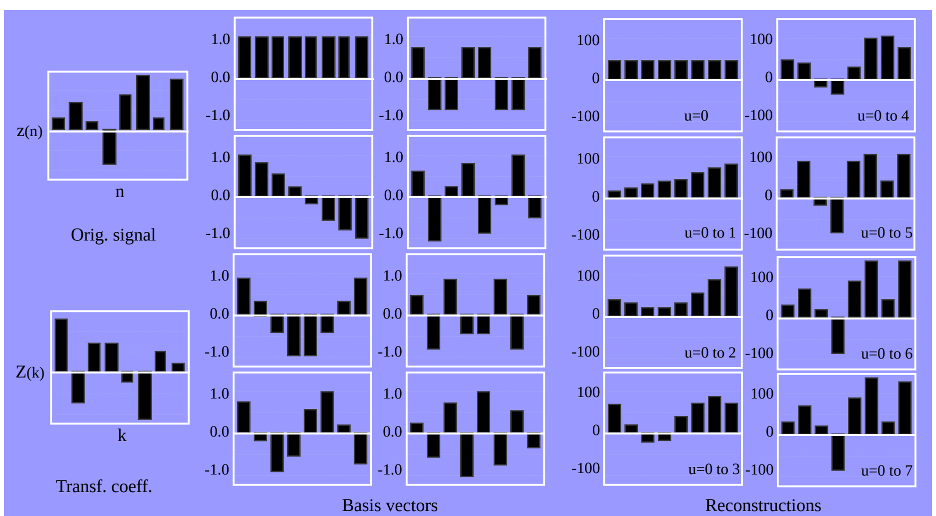
From Ken Lam's DCT talk 2001 (HK Polytech)

Example of 1-D DCT

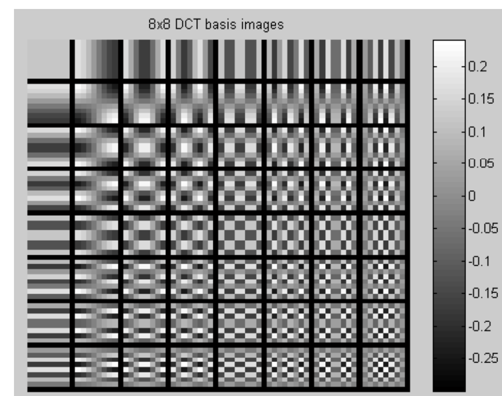


From Ken Lam's DCT talk 2001 (HK Polytech)

Example of 1-D DCT (cont'd)



2-D DCT



- 2-D DCT is a separable transform
  - Apply 1-D DCT to each row, then to each column
- Equivalent to represent an NxN image with a set of orthonormal NxN "basis images"
  - Each DCT coefficient indicates the contribution from (or similarity to) the corresponding basis image

Transform Coding

- Use transform to pack energy to only a few coefficients

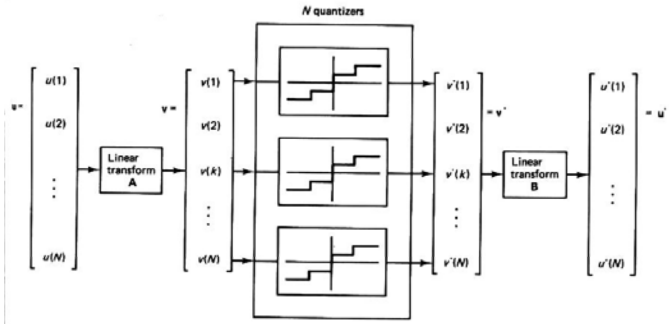


Figure 11.15 One-dimensional transform coding.

From Jain's Fig.11.15

- How many bits to be allocated for each coeff.?
  - Determined by their variance: low freq. coeff. have higher var. (more info.)
    - *More bits for coeff. with high variance* σ_k^2 *to keep total MSE small*
  - Also incorporate perceptual model: fewer bits for high-freq coeff.

Block-based Transform Coding

- Encoder
  - Step-1 Divide an image into *m* x *m* blocks and perform transform
  - Step-2 Determine bit-allocation for coefficients
  - Step-3 Design quantizer and quantize coefficients (lossy!)
  - Step-4 Encode quantized coefficients
- Decoder

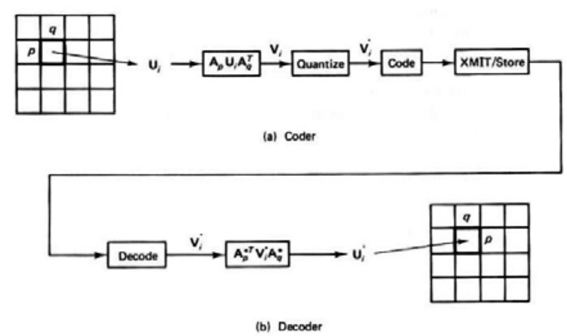
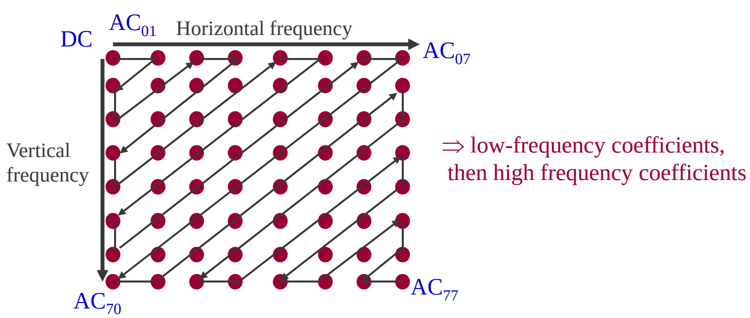


Figure 11.17 Two-dimensional transform coding.

From Jain's Fig.11.17

How to Encode Quantized Coeff. in Each Block?

- Basic tools
  - Entropy coding ~ run-length coding, Huffman, etc.
  - Predictive coding ~ esp. for DC
- Ordering
  - zig-zag scan for block-DCT to better exploit run-length coding gain



Summary: List of Compression Tools

- Lossless encoding tools
  - Entropy coding: Huffman, Lemple-Ziv, and others (arithmetic coding)
  - Run-length coding
- Lossy tools for reducing redundancy
  - Quantization: scalar quantizer vs. vector quantizer
  - Truncations: discard unimportant parts of data
- Facilitating compression via Prediction
  - Encode prediction parameters and residues with less bits
- Facilitating compression via Transforms
  - Transform into a domain with improved energy compaction

Put Basic Tools Together:

JPEG Image Compression Standard

JPEG Compression Standard (early 1990s)

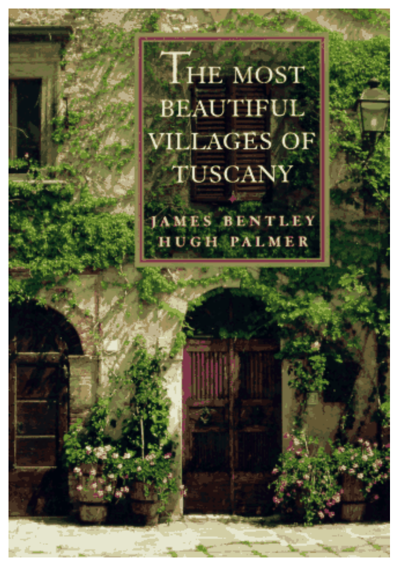
- JPEG - Joint Photographic Experts Group
  - Compression standard of generic continuous-tone still image
  - Became an international standard in 1992
- Allow for lossy and lossless encoding of still images
  - Part-1 DCT-based lossy compression
    - *average compression ratio 15:1*
  - Part-2 Predictive-based lossless compression
- Sequential, Progressive, Hierarchical modes
  - Sequential ~ *encoded in a single left-to-right, top-to-bottom scan*
  - Progressive ~ *encoded in multiple scans to first produce a quick, rough decoded image when the transmission time is long*
  - Hierarchical ~ *encoded at multiple resolution to allow accessing low resolution without full decompression*
- JPEG 2000: based on Wavelet transf. + improved encoding

Baseline JPEG Algorithm

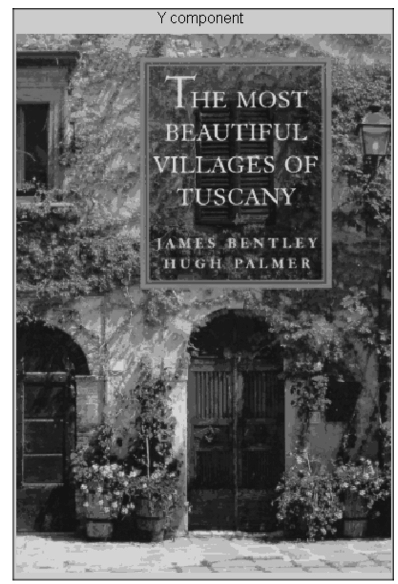
- "Baseline"
  - Simple, lossy compression
    - *Subset of other DCT-based modes of JPEG standard*
- A few basics
  - 8x8 block-DCT based coding
  - Shift to zero-mean by subtracting 128 ➔ [-128, 127]
    - *Allows using signed integer to represent both DC and AC coeff.*
  - Color (YCbCr / YUV) and downsample
    - *Color components can have lower spatial resolution than luminance*
  - Interleaving color components

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(Based on Wang's video book Chapt.1)



475 x 330 x 3 = 157 KB

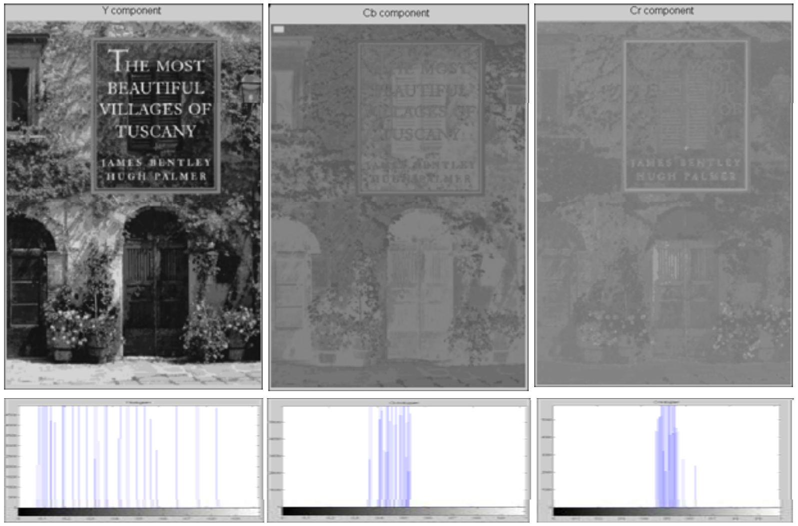


luminance

From Liu's EE330 (Princeton)

Y Cb Cr Components

From Liu's EE330 (Princeton)



Assign more bits to Y, less bits to Cb and Cr

Block Diagram of JPEG Baseline

From Wallace's JPEG tutorial (1993)

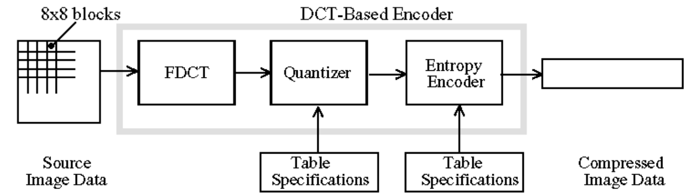


Figure 1. DCT-Based Encoder Processing Steps

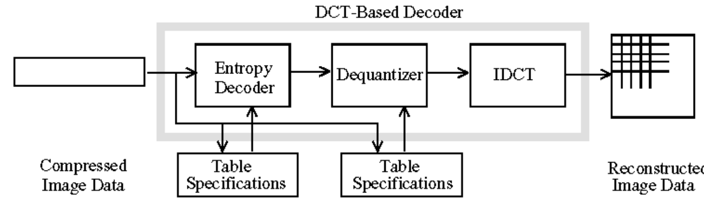


Figure 2. DCT-Based Decoder Processing Steps

Illustration of JPEG Baseline Algorithm

Flash Demo by Dr. Ken Lam (Hong Kong PolyTech Univ.)

[Also posted at course webpage under image project]

Lossless Coding Part in JPEG

- Differentially encode DC
  - (lossy part: DC differences are then quantized.)
- AC coefficients in one block
  - Zig-zag scan after quantization for better run-length
    - *save bits in coding consecutive zeros*
  - Represent each AC run-length using entropy coding
    - *use shorter codes for more likely AC run-length symbols*

Lossy Part in JPEG

- Important tradeoff between bit rate and visual quality
- Quantization (adaptive bit allocation)
  - Different quantization step size for different coeff. bands
  - Use same quantization matrix for all blocks in one image
  - Choose quantization matrix to best suit the image
  - Different quantization matrices for luminance and color components
- Default quantization table
  - "Generic" over a variety of images
- Quality factor "Q"
  - Scale the quantization table
  - Medium quality Q = 50% ~ no scaling
  - High quality Q = 100% ~ quantization step is 1
  - Poor quality ~ small Q, larger quantization step
    - *visible artifacts like ringing and blockiness*

Quantization Table Recommended in JPEG

8x8 Quantization Table for Luminance

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

8x8 Quantization Table for Chrominance

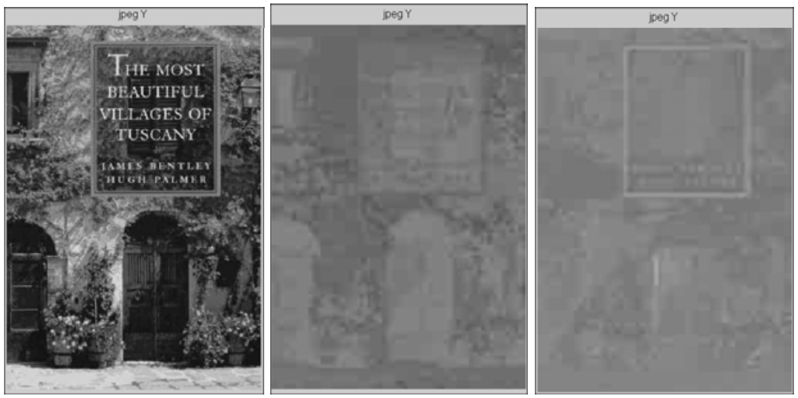
17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

JPEG Compression (Q=75% & 30%)

45 KB 22 KB

From Liu's EE330 (Princeton)

Y Cb Cr After JPEG (Q=30%)



From Liu's EE330 (Princeton)

Uncompr

JPEG 75% (

JPEG 50% (

JPEG 30% (9

JPEG 10% (5KB)

Summary

- Basic tools for compression
  - PCM coding, entropy coding, run-length coding
  - Quantization and truncation
  - Predictive coding
  - Transform coding: DCT-based
- JPEG image compression
  - 8x8 Block-DCT based transform coding
  - Use predictive coding, quantization, run-length coding, and entropy coding
- This week's Lab session:
  - Continued on Design Project 1 => image compression
- Readings (see course webpage)
  - "Data Compression": Sections 7.1 – 7.3, 7.4.1 – 7.4.11, 7.5.1 – 7.5.3
  - JPEG tutorial by Wallace