ENEE631 Fall 2001
Lecture-7

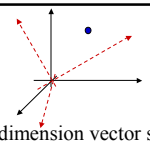
Introduction to Image Compresion

Min Wu

Electrical & Computer Engineering
Univ. of Maryland, College Park

☞ *http://www.ece.umd.edu/class/enee631/*
☞ *minwu@eng.umd.edu*

Review of Last Time



- 1-D transform of a vector
  - Represent an N-sample sequence as a vector in N-dimension vector space
  - Transform
    - *Different representation of this vector in the space via different basis*
    - *e.g., 1-D DFT from time domain to frequency domain*
  - Forward transform
    - *In the form of inner product*
    - *Project a vector onto a new set of basis to obtain N "coefficients"*
  - Inverse transform
    - *Use linear combination of basis vectors weighted by transform coeff. to represent the original signal*
- 2-D transform of a matrix
  - Rewrite the matrix into a vector and apply 1-D transform
  - Separable transform allows applying transform to rows then columns

Review of Last Time (cont'd)

- Representation with orthonormal basis ⇔ Unitary transform
  - Preserve energy, decorrelation, etc.
- Common unitary transforms
  - DFT, DCT, KLT
- Which transform to choose?
  - Depend on need in particular task/application
  - DFT ~ reflect physical meaning of frequency or spatial frequency
  - KLT ~ optimal in energy compaction
  - DCT ~ real-to-real, and close to KLT's energy compactability
  - Today's addition: Haar Transform

Construction of Haar functions

power of 2

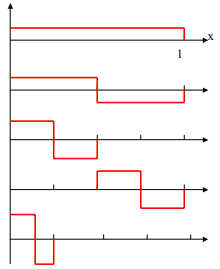
$k = 2^p + q - 1$

"reminder"

- Unique decomposition of integer $k \Leftrightarrow (p, q)$
  - k = 0, …, N-1 with $N = 2^n$, 0 <= p <= n-1
  - q = 0, 1 (for p=0); 1 <= q <= 2^p (for p>0)
  - e.g., k=0 ⇔ (0,0), k=1 ⇔ (0,1); k=2 ⇔ (1,1), k=3 ⇔(1,2)
- $h_k(x) = h_{p,q}(x)$ for x ∈ [0,1]

$$h_0(x) = h_{0,0}(x) = \frac{1}{\sqrt{N}} \quad \text{for } x \in [0,1]$$

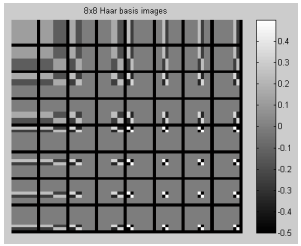
$$h_k(x) = h_{p,q}(x) = \begin{cases} \frac{1}{\sqrt{N}} 2^{p/2} & \text{for } \frac{q-1}{2^p} \le x < \frac{q-\frac{1}{2}}{2^p} \\ -\frac{1}{\sqrt{N}} 2^{p/2} & \text{for } \frac{q-\frac{1}{2}}{2^p} \le x < \frac{q}{2^p} \\ 0 & \text{for other } x \in [0,1] \end{cases}$$



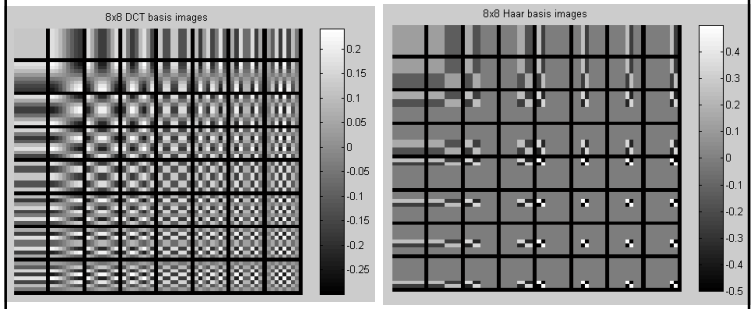
Haar Transform

- Haar transform *H*
  - Sample $h_k(x)$ at {m/N}
    - *m = o, ..., N-1*
  - Real and orthogonal
  - Transition at each scale *p* is localized according to *q*

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$



Compare Basis Images of DCT and Haar



- NxN DCT basis images reflect transitions throughout the NxN image
  - Ordered by frequency
- Haar basis images can reflect local transitions
  - Ordered both by frequency and by spatial location index

Image Compression

Part-1. Basics

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
  - 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

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

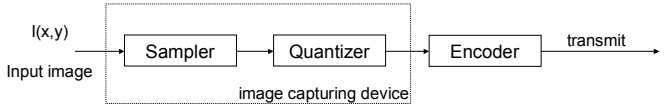
Example: Storing An Encyclopedia

- 500,000 pages of text (2kB/page) ~ 1GB => 2:1 compress
- 3,000 color pictures (640×480×24bits) ~ 3GB => 15:1
- 500 maps (640×480×16bits=0.6MB/map) ~ 0.3GB => 10:1
- 60 minutes of stereo sound (176kB/s) ~ 0.6GB => 6:1
- 30 animations with average 2 minutes long (640×320×16bits×16frames/s=6.5MB/s) ~ 23.4GB => 50:1
- 50 digitized movies with average 1 minute long (640×480×24bits×30frames/s = 27.6MB/s) ~ 82.8GB => 50:1

➔ Require a total of 111.1GB storage capacity if without compression
➔ Reduce to 2.96GB if with compression

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: companding, contrast quantization, dithering, etc.
    - *Halftone use 1bpp but usually upsampling ~ saving less than 2:1*



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 treat*
  - If use less bits for likely values "0" ~ Variable Length Codes (VLC)
    - *"0" => [0], "1" => [10], "2" => [110], "3" => [111]*
    - *Use 1.75 bits on average ~ saves 0.25 bpp!*
- Bring probability into the picture
  - Previously use prob. distr. to reduce MSE of quantization
  - Now use prob. distr. to reduce average # bits per quantized sample

Entropy coding

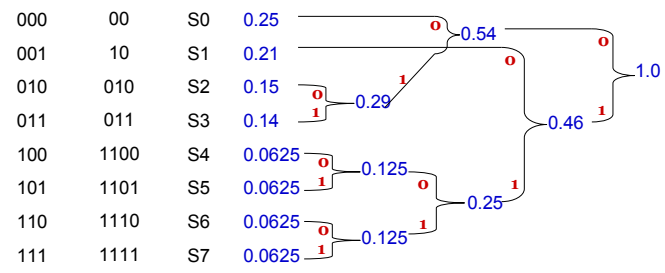
- Idea: use less bits for commonly seen values
- How many # bits needed?
  - Limit of compression => "Entropy"
    - *Measures the uncertainty or amount of avg. information of a source*
    - *Definition:* $H = \sum_i p_i \log_2 (1 / p_i)$ bits
      - e.g., entropy of previous example is 1.75
    - *Can't represent a source perfectly with less than avg. H bits per sample*
    - *Can represent a source perfectly with avg. H+ε bits per sample (Shannon Lossless Coding Theorem)*
  - "Compressability" depends on the sources
- Important to decode coded stream efficiently without ambiguity
- See info. theory course for more theoretical details

E.g. of Entropy Coding: Huffman Coding

- Variable length code
  - assigning about $\log_2 (1 / p_i)$ bits for the i^{th} value
    - *has to be integer# of bits per symbol*
- Step-1
  - Arrange p_i in decreasing order and consider them as tree leaves
- Step-2
  - Merge two nodes with smallest prob. to a new node and sum up prob.
  - Arbitrarily assign 1 and 0 to each pair of merging branch
- Step-3
  - Repeat until no more than one node left.
  - Read out codeword sequentially from root to leaf

Huffman Coding (cont'd)

000	00	S0	0.25
001	10	S1	0.21
010	010	S2	0.15
011	011	S3	0.14
100	1100	S4	0.0625
101	1101	S5	0.0625
110	1110	S6	0.0625
111	1111	S7	0.0625
PCM	Huffman		



Huffman Coding: Pros & Cons

- Pro
  - Simplicity in implementation (table lookup)
  - For a given block size Huffman coding gives the best coding efficiency
- Con
  - Need to obtain source statistics
- Improvement
  - Code a group of symbols as a whole to allow fractional # bit per symbol
  - Arithmetic coding
  - Lempel-Ziv coding or LZW algorithm
    - *"universal", no need to estimate source statistics*

Discussion: Coding a Binary Image

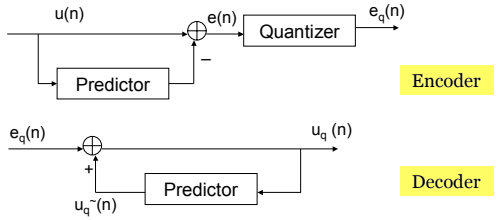
- How to efficiently encode it?
  - " 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 …"
- Run-length coding (RLC)
  - Code length of runs of "0" between successive "1"
    - *run-length of "0" ~ # of "0" between "1"*
    - *good if often getting frequent large runs of "0" and sparse "1"*
  - E.g., => (4) (0) (3) (1) (6) (0) (0) … …
  - Assign fixed-length codeword to run-length
  - Or use variable-length code like Huffman to further improve
- RLC Also applicable to general binary data sequence with sparse "1" (or "0")

Probability Model of Run-Length Coding

- Assume successive "0" occur independently w.p. *p*
- Prob. of getting an *L* runs of "0" *(L=0,1, …, M)*
  - $P(L) = p^L (1-p)$ for $0 <= l <= M-1$ (geometric distribution)
  - $P(L) = p^M$ for $L=M$
- Avg. # binary symbols per run
  - $S_{avg} = \Sigma_L (L+1) p^L (1-p) + M p^M = (1 - p^M) / (1 - p)$
  - Compression ratio $C = S_{avg} / \log_2 (M+1) = (1 - p^M) / [(1-p) \log_2(M+1)]$
- Example
  - P = 0.9, M=15 ➔ S_{avg} = 7.94, C = 1.985, H = 0.469 bpp
  - Avg. run-length coding rate B_{avg} = 4 bit / 7.94 ~ 0.516 bpp
  - Coding efficiency = H / B_{avg} ~ 91%

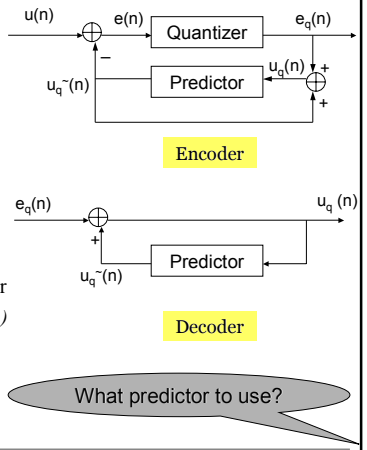
How to Encode Correlated Sequence?

- 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



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: *Differential PCM (DPCM)*
  - Use quantized sequence $u_q(n)$ for prediction at both encoder and decoder
    - $u_q\tilde{}(n) = f(u_q(n-1), u_q(n-2), \ldots, u_q(n-m))$
  - Prediction error $e(n)$
  - Quantized prediction error $e_q(n)$
  - Distortion $d(n) = e(n) - e_q(n)$



Summary

- Finish image transform
- Begin basics on image coding/compression
  - PCM coding
  - Entropy coding
    - *Huffman*
    - *Run-length*
  - Predictive coding
- Next time: transform coding

Assignment

- Readings
  - Jain's 11.1-11.3
- Reference
  - Bovik's Handbook Sec.5.1
- "Food for thoughts"
  - What predictors to use for image predictive coding?
    - *pros and cons*
- Announcement
  - 1st in-class exam will be given after "Image Compression"

ENE631 Course Project: Introduction

Topic: Video Coding and Processing

- Team project
  - 2 students per team
  - A few 3-person teams are allowed by instructor's permission
- What to turn in finally?
  - A video codec (encoder-decoder)
  - Self-proposed part on video processing/analysis

Step-by-Step Approach

- [9-10/01] 3 Building Blocks (BB) ~ similar to "assignment"
  - individual work
  - should keep a big picture of the project in mind
- [11/2001] Team-up and submit project proposal
- [12/2001] Project demo, presentation, and report
  - peer review
- Details will be announced soon
- First BB on image transform will be posted this weekend