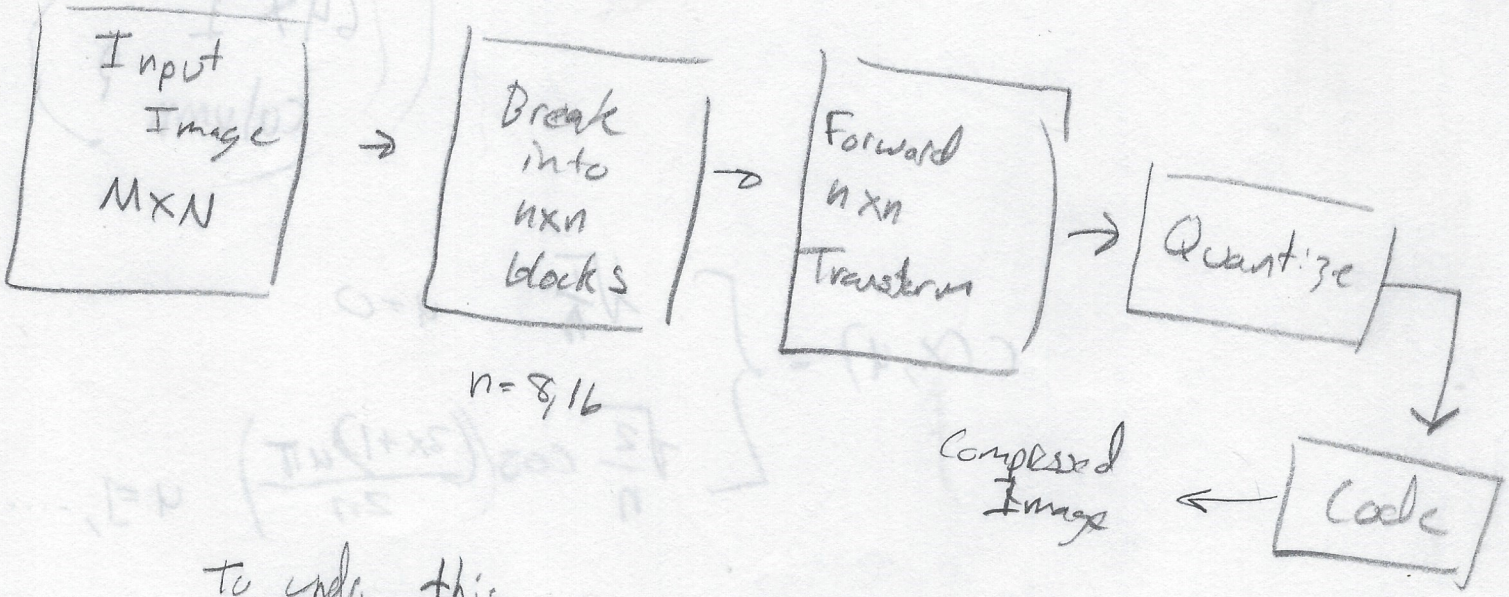
Digital Image Processing

Lecture 16 Lossy Image Compression

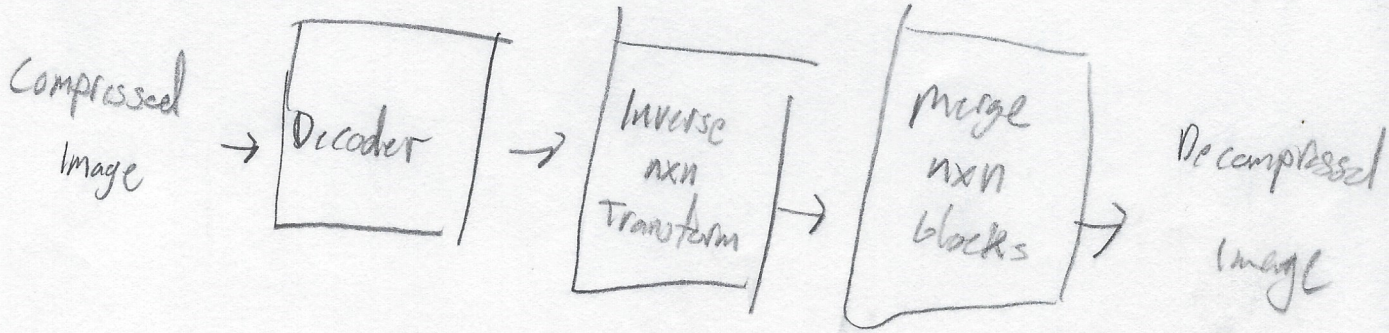
Lossy Image CODING

Key Concept: Throw away perceptually Insignificant Information to get Coding Gains

Block Transform CODING



To undo this



(2)

JPEG uses the Discrete Cosine transform (DCT)

a) IT is Real

b) IT compacts energy into Low frequencies

$f(x,y)$ → [DCT] → $T(u,v)$

8×8 block → 8×8 block

$$T(u,v) = \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y)\, c(x,u)\, c(y,v)$$

64 x 1 column ?

$$c(x,u) = \begin{cases} \sqrt{\frac{1}{n}} & u = 0 \\ \sqrt{\frac{2}{n}} \cos\left(\frac{(2x+1)u\pi}{2n}\right) & u = 1, \ldots 7 \end{cases}$$

Get (Lossy) Compression By

Quantizing / Allocating Bits To each value of $T_{u,v} \rightarrow \hat{T}_{(u,v)}$

- Zonal Coding: Depending on Location in 8x8 block use more or less bits.

~~On The~~

MatLab

Zonal

Sum(Zonal(:))/64

= 1.9063 almost 2 bits per pixel

On The average, use < 8 bits per pixel

8 7 6 5
7 6 5
6 5
5

1
1 1
0 0 0

(4)

Alternative Threshold Coding $T(u,v)$

a) Select m Largest Coefficients in T and Code these

b) Select Coefficients that account for $p\%$ of The total Energy

c) Select all Coefficients whose magnitude is above τ

ALL of These adapt to block-by-block Contents

Disadvantage: Must also Send Locations of Coded Coefficients per block → BITRATE ↑

JPEG Algorithm uses a "normalization matrix"

To Specify Quantization

$$Z(u,v) = \begin{bmatrix} 16 & 11 & 10 & \text{ETC.} \\ 12 & 12 & 14 & \\ 14 & 13 & 16 & \\ & & & \\ \text{ETC.} & & & \ddots \end{bmatrix}$$

$$\begin{bmatrix} \text{small} & \text{med} \\ \text{med} & \text{large} \end{bmatrix}$$

$$\begin{bmatrix} 104 & 113 & 92 \\ 121 & 120 & 101 \\ 110 & 103 & 99 \end{bmatrix}$$

How to use $Z(u,v)$ to Quantize $T(u,v)$?

$$\hat{T}(u,v) = \text{round}\left(\frac{T(u,v)}{Z(u,v)}\right)$$

$$\begin{bmatrix} 16 & & \\ & 29 & \\ & & 120 \end{bmatrix}$$

$Z(u,v)$

$$\bigcirc \rightarrow \hat{T} = \text{round}\left(\frac{50}{16}\right) = 3$$

$$\square \rightarrow \hat{T} = \text{round}\left(\frac{50}{29}\right) = 2$$

$$\triangle \rightarrow \hat{T} = \text{round}\left(\frac{50}{120}\right) = 0$$

To reconstruct

$$\bar{T}(u,v) = Z(u,v)\,\hat{T}(u,v)$$

$$\bigcirc \Rightarrow 3 \cdot 16 = 48$$

$$\square \Rightarrow 2 \cdot 29 = 58$$

$$\triangle \Rightarrow 0 \cdot 120 = 0$$

(b) After Coefficients are Quantized,

Arrang them into a List of 64 numbers,

using a Zig Zag ORDER

IDEA: Zig-Zag will be ALL 0 After some point.

Last step: Convert STRING to Binary and Basically Huffman Code The AC Coefficents

DC Coefficient is Coded with respect to Difference from previous DC.

Demo on the HW6

MatLab Procedure (10.41)

(8)

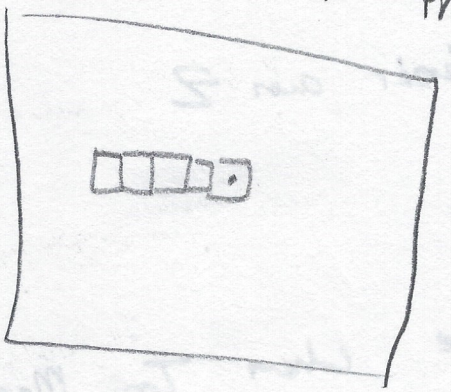
Notes on JPEG

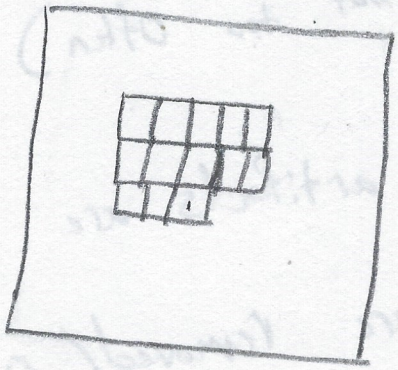
- To vary JPEG "Quality", multiply the fixed Z matrix by a smaller # (Higher Quality) or a bigger # (Lower quality)
- USER can also specify their own Z (Not too often)
- blocking artifacts are visible when too many coeffs are removed/ Q'D
- For Color Images, RGB → L | UV
  Luminance Chrominance
  UV, coded at much lower bitrate than L.

(10)

Predictive Coding: Exploit the fact that there's a lot of correlation between Neighboring Pixels

IDEA: Predict Pixel's value from its neighbors, Then code the Error in Prediction

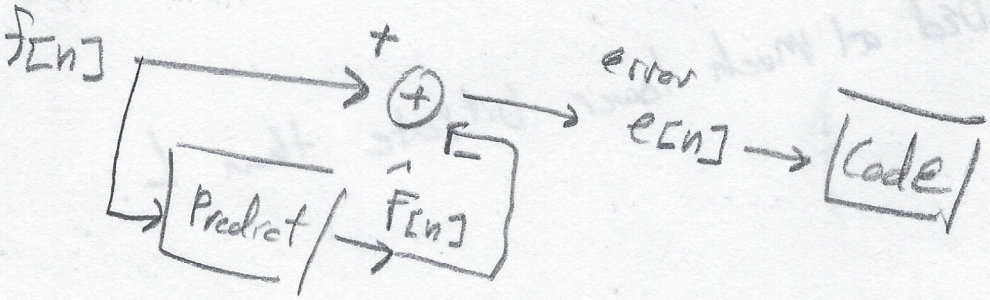




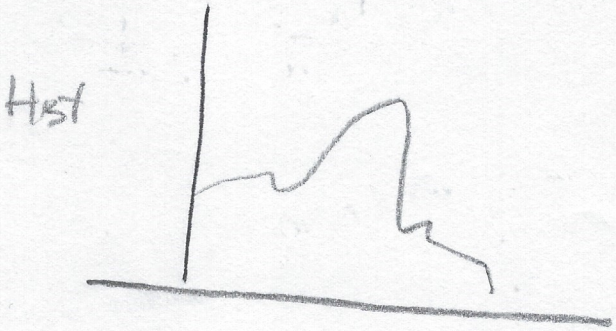
my Predicted Value could be

AR model

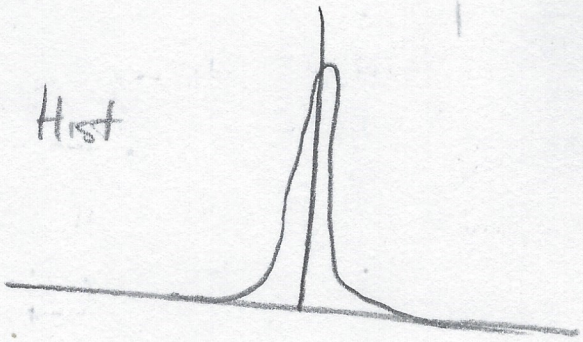
$$\hat{f}(x,y) = \sum_{i=1}^{M} \alpha_i \, f(x, y-i)$$



Relies on the Difference in Histograms



original Histogram

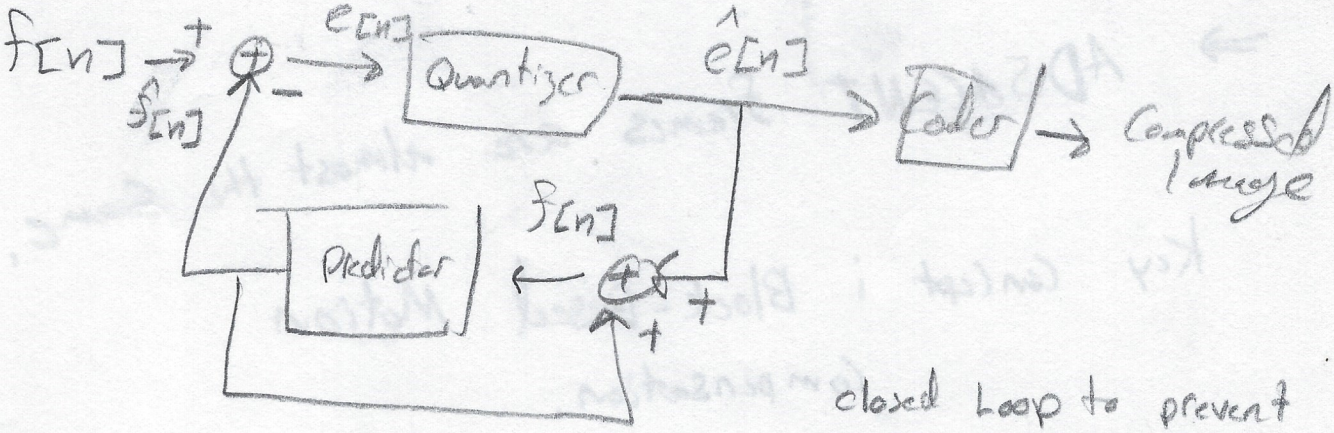


Error Histogram e[n]

Lower Entropy

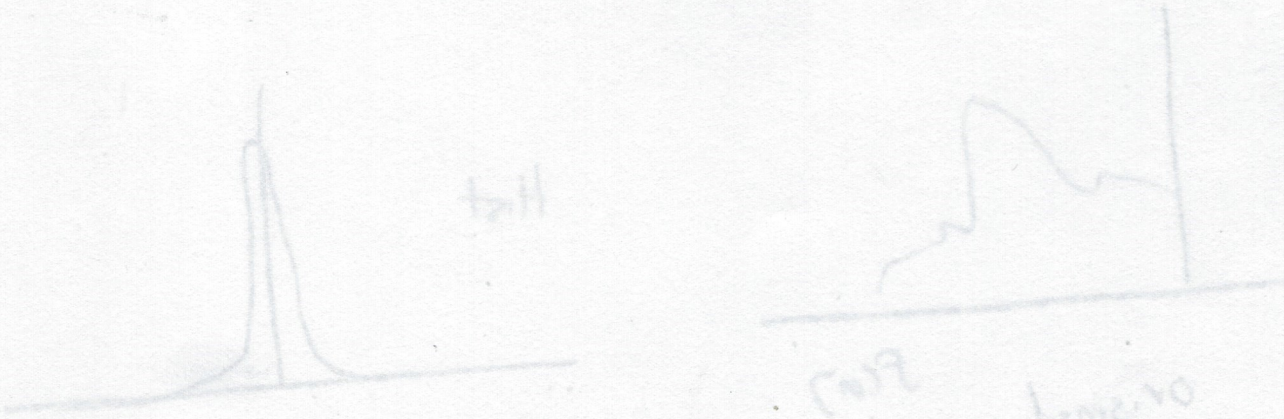
→ more predictable (Natural Images)

For Lossy compression, we would need to include the Quantizer inside the Predictor



closed Loop to prevent error accumulation at output

block by block compression



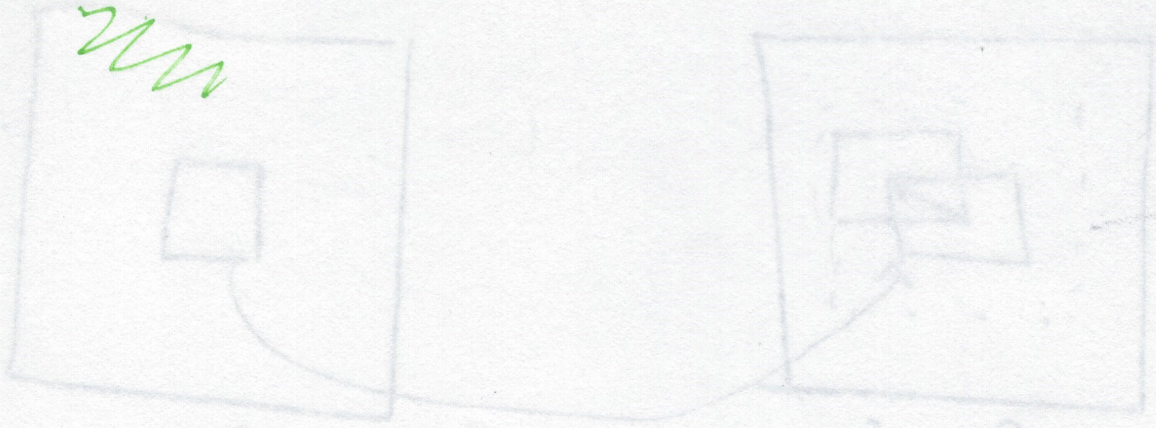
Predictive Coding is Critical for Video Compression

IMAGES ARE SPACED Extremely Closely In Time (e.g., 1/30 Sec Apart)

⇒ ADJACENT frames are almost the same.

Key Concept: Block-Based Motion Compensation.

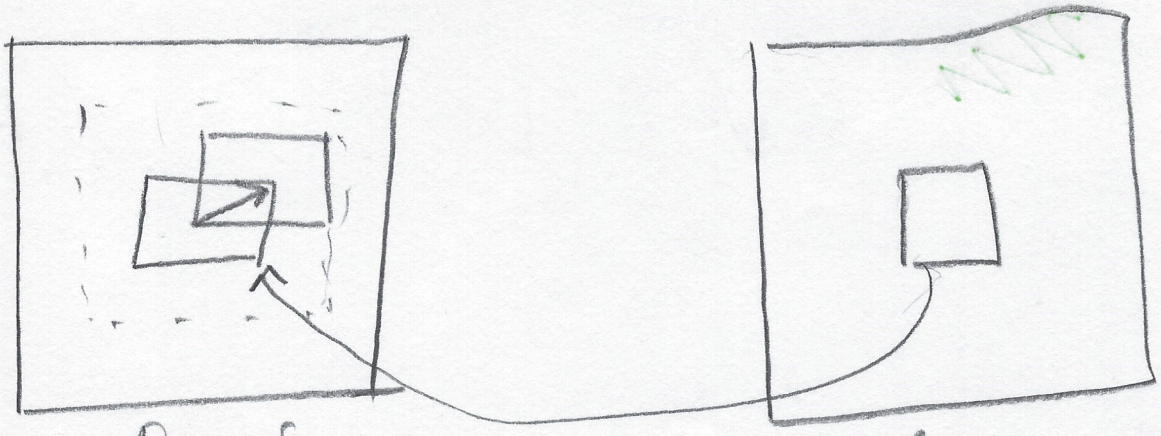
MPEG USED 3 Types of frames



I frames: "Independently Coded" or "INTRA"
Code These Independently with JPEG.

P frames: "Predictive" frames – Predict
Macro blocks Based on Previous P or I
(Sum of Squared Differences)
(Send Arrow to best Match + Code error)

B frames: "Bidirectional" – Can predict Based on P/I
from either side



For each Macro block,

- Search for the best Match in Prev frame within a reasonable Neighborhood (dx, dy)
- To Code, Send $(dx\,dy)$ and Quantize the Residual

$$e(x,y) = f_i(x,y) - f_{i-1}(x+dx, y+dy)$$

Should Be ≈ 0 Most of the TIME