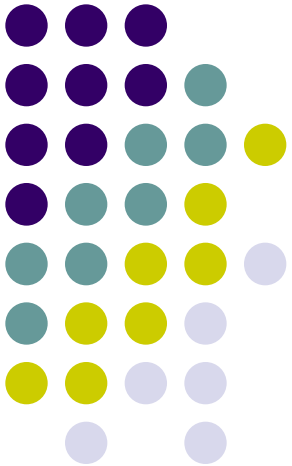
Introduction to Media Computing

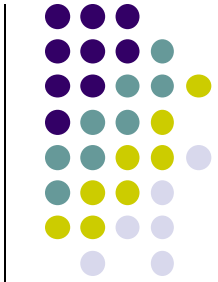
Lecture 10: Image Filters &Transformation



Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

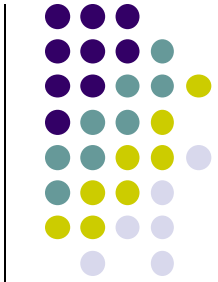
TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

Summary



- Introduction to Media Computing
- Introduction to Text Retrieval
- Image Representation and Retrieval
- Multimedia Indexing and Search Architecture
- Fundamentals of Digital Image
- Introduction to Digital Audio
- **Image Filters and Transformation**

Outline of Lecture

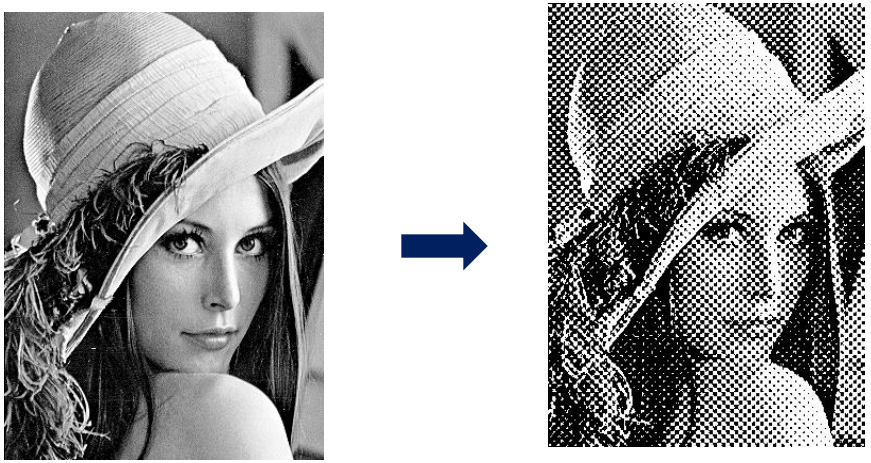


- Dithering
- Image Filters
- Color Transfer between Images
- Image Filtering App

- REFERENCES:
  - Z.N. Li and M.S. Drew (2004). Fundamentals of Multimedia. Prentice Hall (Chapter 3)
  - J. Burg (2009). The Science of Digital Media. Pearson/ Prentice Hall. (Chapter 3)
  - Erik Reinhard et al. Color Transfer between Images. Applied Perception, Sep/Oct 2001, pp 34-41

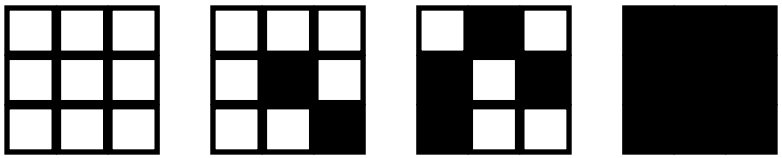
Dithering

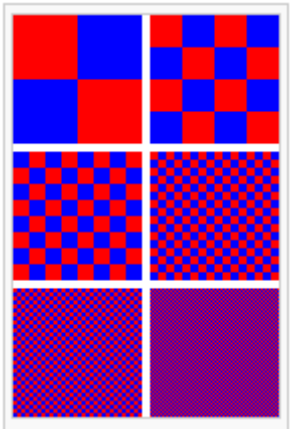
- Techniques for simulating colors that are not available in a palette by using the available colors that are blended by the eye so that they look like the desired colors
  - Especially useful when you change from RGB to indexed color
  - For example, to print a 8-bit gray–scale image on a 1-bit printer; or to display a 24-bit color on 8-bit display.
  - Basic strategy is to trade intensity resolution for spatial resolution



Dithering -2

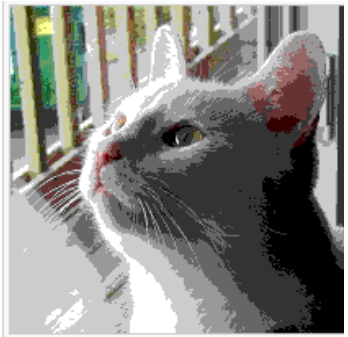
- Key is how you simulate the colors/shades and “fool” the eye with less available colors
  - Black/white dithering to achieve gray scale – simulate 10 scale of grays





  - Take advantage of the human eye's tendency to "mix” two colors in close proximity to one another
  - Examples:

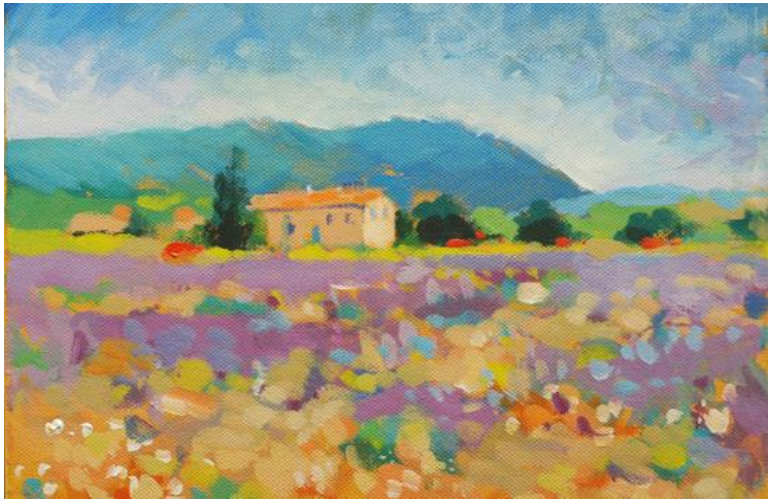
Original, 2^{24} colors



no dithering, 2^{8} colors

with dithering, 2^{8} colors

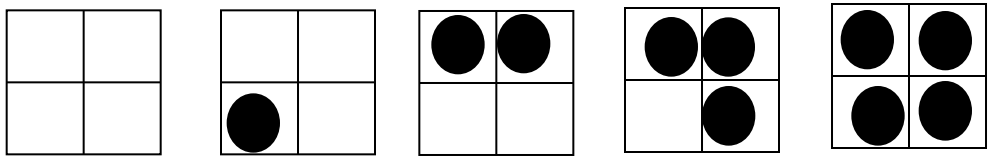
Dithering Effects used Effectively in Impressionist Painting



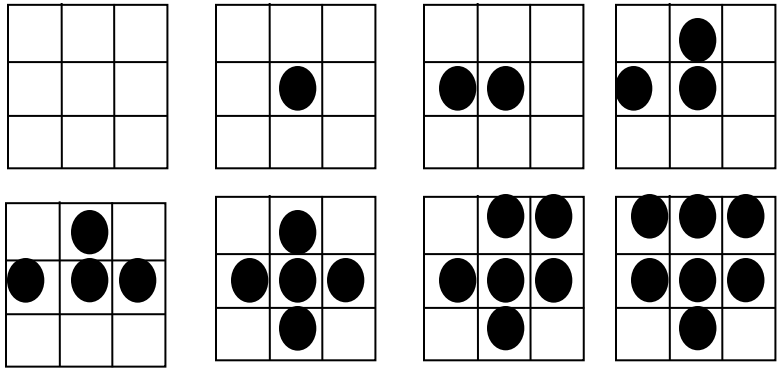
Half-Tone Printing -1

- One strategy in used in printing industry:
  - Replace a pixel value by a larger pattern, say 2 x 2 or 4 x 4, such that the number of printed dots approximates the varying-intensity of original pixel
  - For example, if we use a 2 x 2 dither to give 5 (n^2+1) patterns:

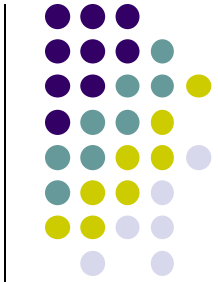
2x2 Halftone Patterns are:



  - Or 4 x 4 patterns:



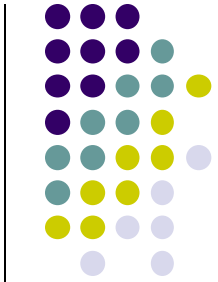
Half-Tone Printing -2



- Strategy for selection of patterns
  - Key strategy is to avoid Symmetry
  - At each level we select all the pixels on the previous level and turn on one pixel more
  - Black circles should be as close as possible to the center of the pattern

- Process:
  - Re-map image values in 0..255 into the new range 0.. (n^2-1): (integer divide) v / {256/(n^2-1)}
  - Map the resulting value to one of the above (n^2-1) patterns
  - Note that the image size may be much larger, for a dithered image, since replacing each pixel by a n x n array of dots, makes an image n^2 times as large

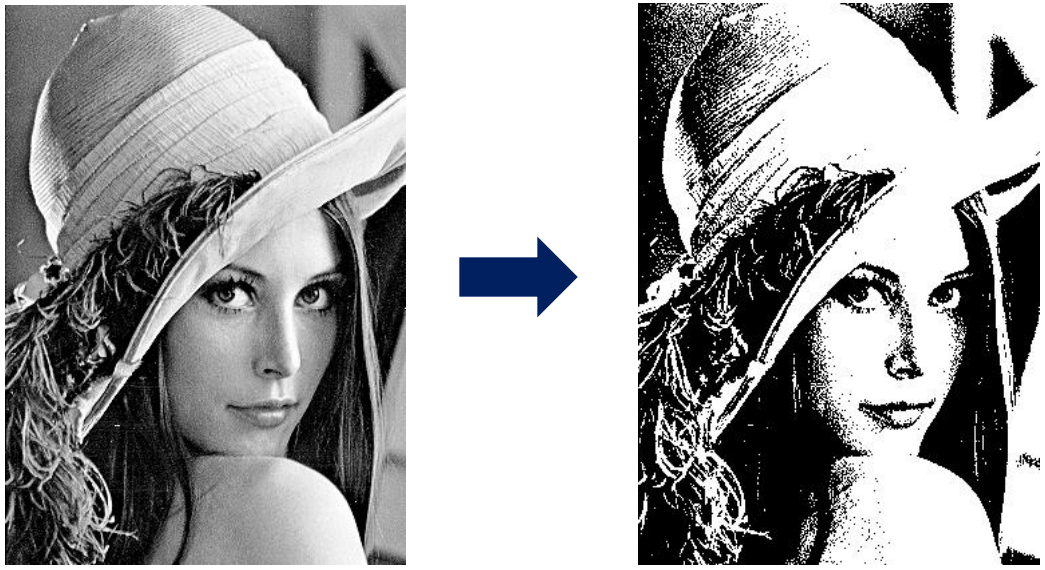
Dithering



- Dithering refers to techniques for approximating halftone without reducing resolution
  - Generate intensity variation with 1-1 mapping from the original image points to the display pixels
  - Achieve simulation with randomizing effects
  - Need to determine how to replace a pixel in higher color-range in original image by a pixel in the target palette with lower color resolution (0/1, or 1 of 256 colors)
- Dithering algorithms for 8-bit image to 1-bit output:
  - Average (thresholding)
  - Random
  - Ordered
  - Floyd-Steinberg

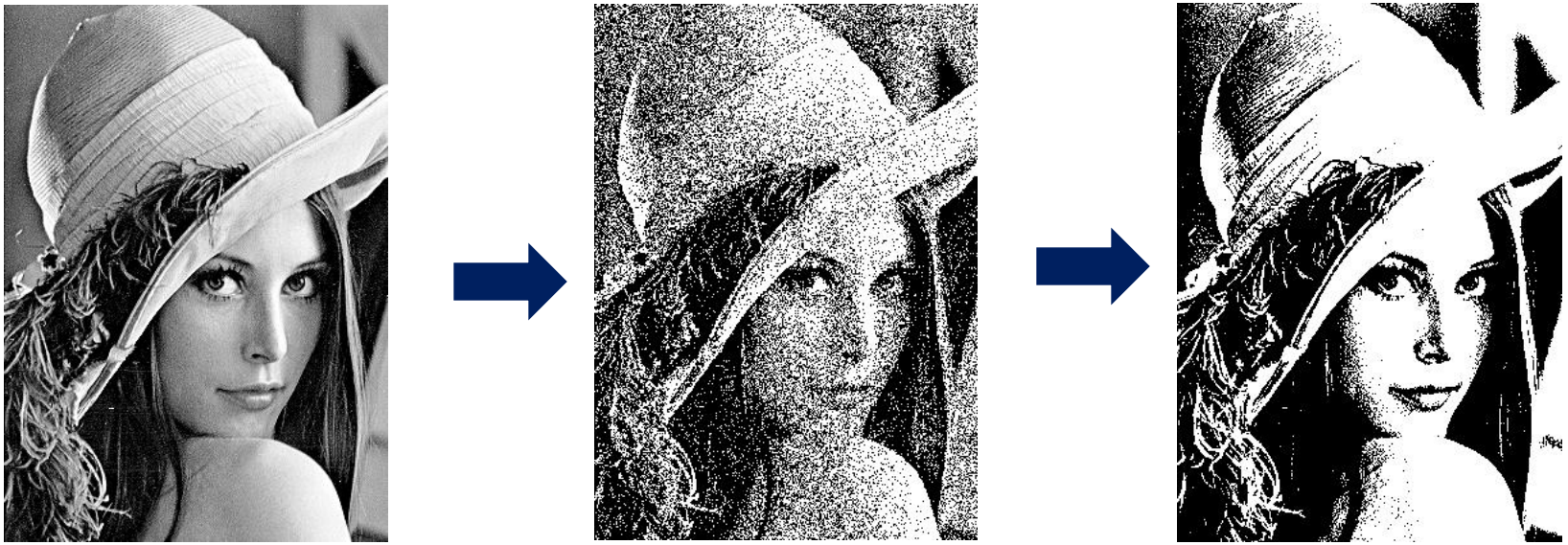
Average (Thresholding) Dithering

- Calculate an average pixel value for image, *Avg*
- For each pixel, if pixel *value* > *Avg*, then pixel = white, otherwise black
- Crude and contrasty, looks patchy (too regular!!)
- Almost never used



Random (Noise) Dithering

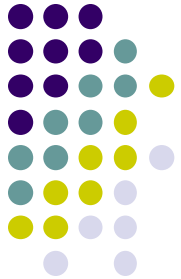
- Aim to eliminate patchiness by adding high frequency noise
- For each pixel, generate a random number *r* (0-255 or 2^n-1)
- If original pixel *value* > *r*, then pixel = white, otherwise black
- Crude and noisy, but looks better than averaging?



Original Random Averaging

- Variations: not inserting noise with every pixel – another random process?

Ordered (Pattern) Dithering -1



- Instead of random dithering at pixel level, do it at block level (or apply convolution)
  - Design n x n array of between 0 – (2^n-1) values
  - Numbers in the Dither matrix determines the appearance of pattern
  - Dither matrix is devised with a simple rule: (a) fill each slot with a successive integer starting from 0; and (b) reorder them such that the average distance between two successive numbers in the matrix is as large as possible, ensuring that the table "wraps" around at edges
  - Similar strategy as in design of half-tone patterns

D_1=

0	2
3	1

D_2=

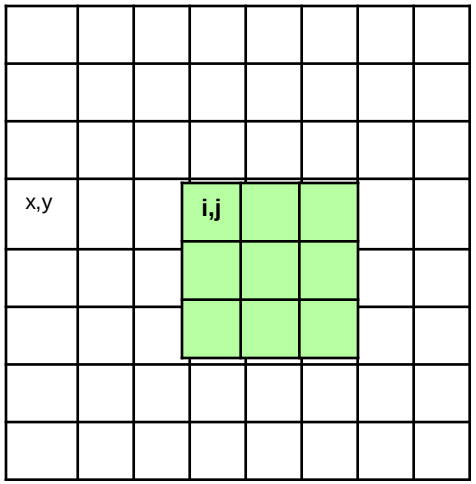
2	6	4
5	0	1
8	3	7

D_3=

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

Ordered Dithering -2

- Ordered Dithering process:
  - Divide the image into ordered cells of size n
  - For each cell, turn on the output bit for a pixel if the intensity level is greater than the corresponding matrix element at that pixel position



- The algorithm:

BEGIN

for $x = 0$ to x_{max} // columns

for $y = 0$ to y_{max} // rows

$i = x \bmod n$

$j = y \bmod n$

// $I(x, y)$ is the input, $O(x, y)$ is the output, D is the dither matrix

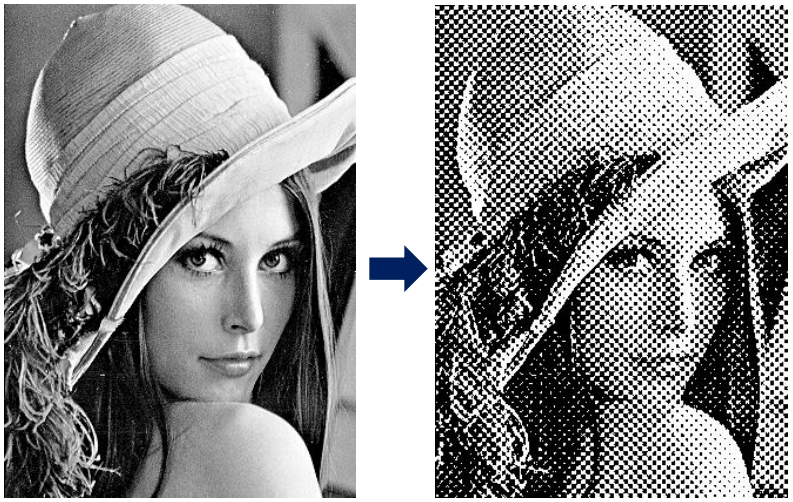
if $I(x, y) > D(i, j)$ $O(x, y) = 1$;

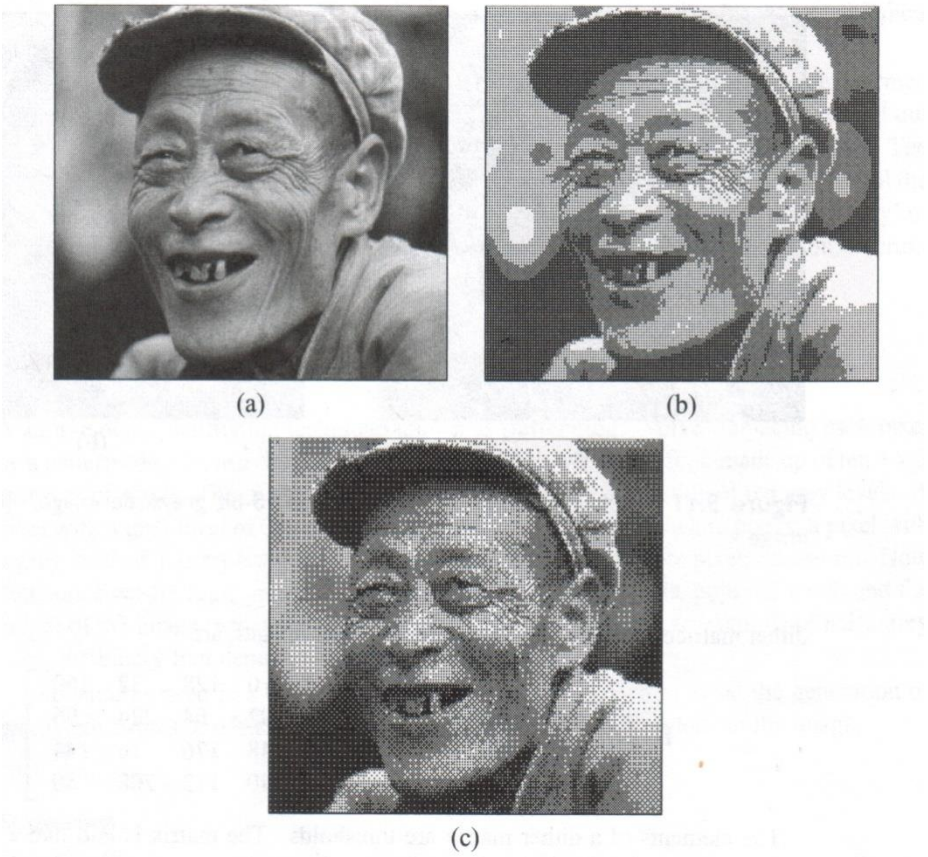
else $O(x, y) = 0$;

END

Ordered Dithering -3

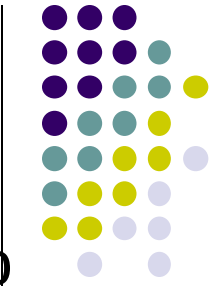
- The results:



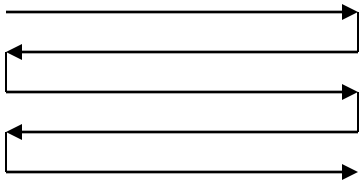


(a) Input image.
(b) Dithered image using D_1.
(c) Dithered image using D_3

(Error Diffusion) Floyd Steinberg Dithering -1



- Earlier methods ignore the errors, this method tries to disperse the errors to neighboring pixels that have not been visited
  - Visit the pixels from top to bottom in a zig-zag

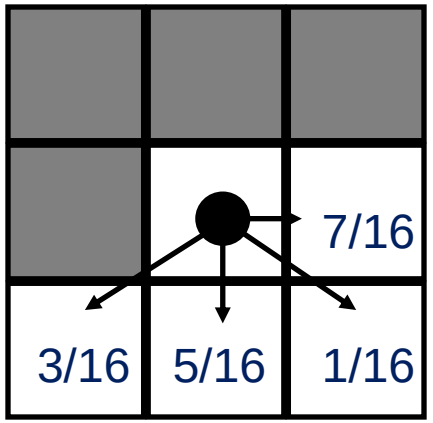


- Threshold each pixel (e.g. using average dither)
- Compute the error at that pixel:
  - Error = the difference between this pixel and the original
  - If you made the pixel 0, *Error* = original;
    if you made it 1, *Error* = original-1

Floyd Steinberg Dithering -2

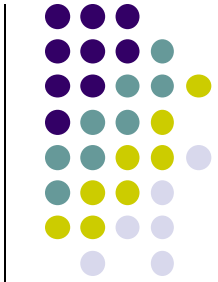
- Propagate error to neighbors by adding some proportion of the error to each unprocessed neighbor
  - A *mask* tells you how to distribute the error
  - The diffusion coefficients have the property that if the original pixel values are exactly halfway in between the nearest available colors, the dithered result is a checkerboard pattern.

    For example 50% grey data could be dithered as a black-and-white checkerboard pattern.



- Easiest to work with floating point image
  - Convert all pixels to 0-1 floating point
- After the errors have been distributed, the pixels are processed a second time
  - To re-align the numbers to target range..

Floyd Steinberg Dithering -3



- A Pseudo code (from wikipedia)

for each y from top to bottom

for each x from left to right

oldpixel := pixel[x][y]

newpixel := find_closest_palette_color(oldpixel)

pixel[x][y] := newpixel

quant_error := oldpixel – newpixel

pixel[x+1][y] := pixel[x+1][y] + quant_error * 7/16

pixel[x-1][y+1] := pixel[x-1][y+1] + quant_error * 3/16

pixel[x][y+1] := pixel[x][y+1] + quant_error * 5/16

pixel[x+1][y+1] := pixel[x+1][y+1] + quant_error * 1/16

- May perform a simple rounding for find_closest_palette_color():

find_closest_palette_color(oldpixel) = floor(oldpixel / 256)

Floyd Steinberg Dithering -3

- The results:

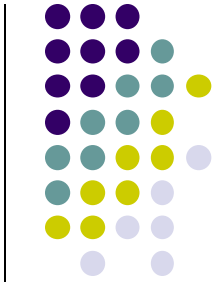
Original image

Floyd Steinberg

Ordered Dithering

- See Demo: http://www.unitzeroone.com/examples/ImageDithering/index.html

Color Dithering -1



- The same techniques can be applied, with some modification, to colors
- Example is Floyd-Steinberg:
  - Apply to each color channel in turn and combine the results
  - Use uniform color table
  - Error is the difference from nearest color in the color table
  - Error propagation is the same as that for greyscale
    - Each color channel treated independently

Color Dithering -2

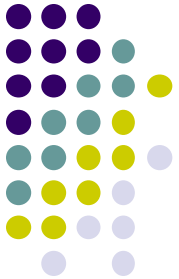
- The Results:
  - Use uniform color table

Original image

Color Dithered Image

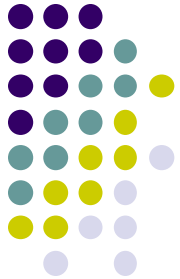
Uniform Quantization

Outline of Lecture

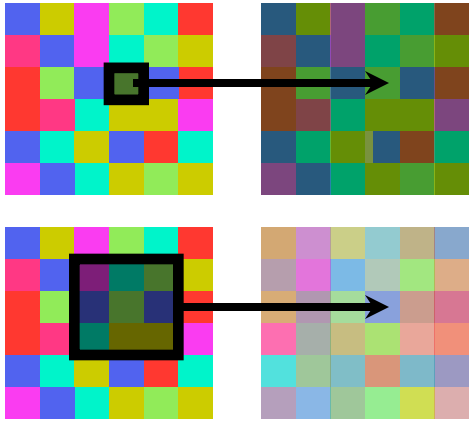


- Dithering
- Image Filters
- Color Transfer between images
- Image Filtering App

Image Filtering



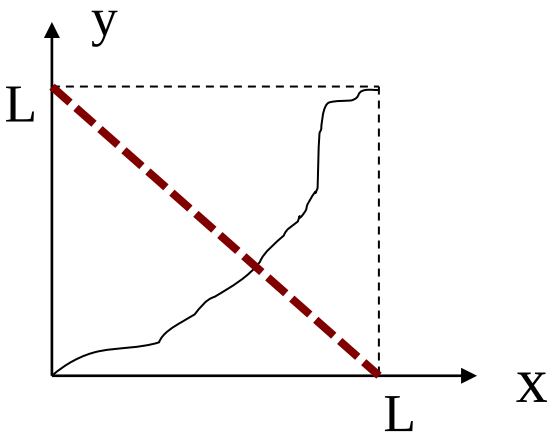
- Aims: Transform contents of images to other (better?) forms
- Image as a two-dimensional signal
  - Point processing: modify pixels independently
  - Filtering: modify based on neighborhood
  - Dithering: is an example of image transform



- Related topics (not in this lecture or this course)
  - Image enhancement and restoration

Point Processing Overview

- Point operations are zero-memory operations where a given gray level $x \in [0,L]$ is mapped to another gray level $y \in [0,L]$ according to a transformation: $y = f(x)$
  - f transforms each pixel value separately (pixel level)
  - Useful for contrast adjustment



L=255: for grayscale images

For example:

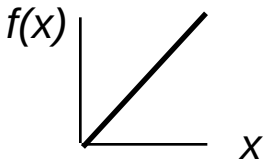
- Different function y=f(x) will lead to different transform

Point Processing

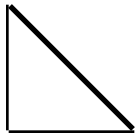
- Input: *x*, Output *y* = *f(x)*
- Suppose our picture is grayscale (a.k.a. monochrome).

  Let *x* denote pixel value, suppose it's in the range [0,1].

f(x) = x identity; no change

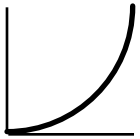


f(x) = 1-x negate an image
(black ←→ white)



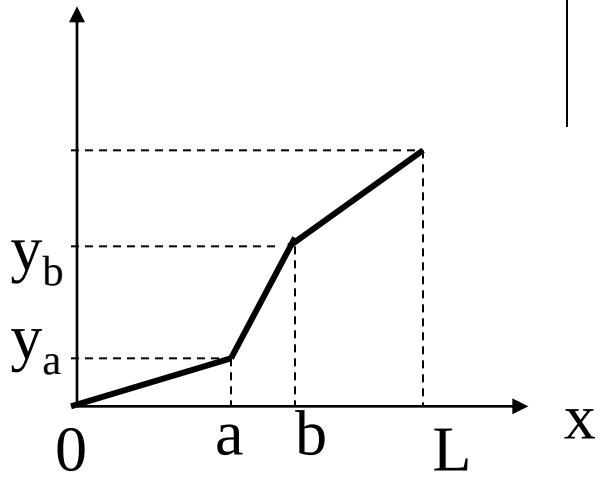
f(x) = x^p, p<1 brighten

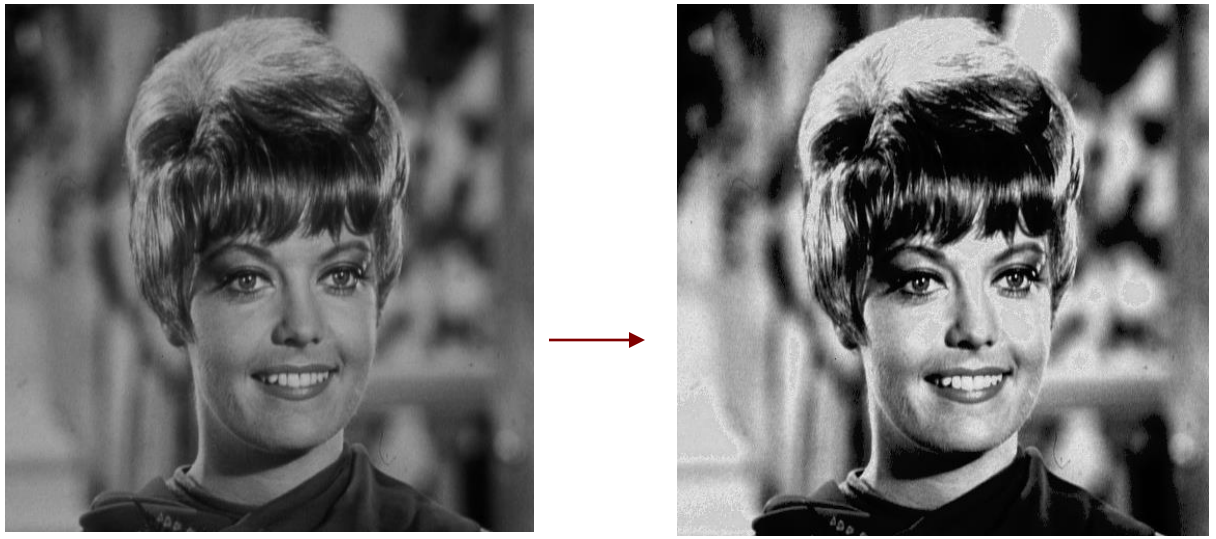
f(x) = x^p, p>1 darken



What about Other Transforms?
Contrast Stretching

$$y = \begin{cases} \alpha x & 0 \le x < a \\ \beta(x-a) + y_a & a \le x < b \\ \gamma(x-b) + y_b & b \le x < L \end{cases}$$



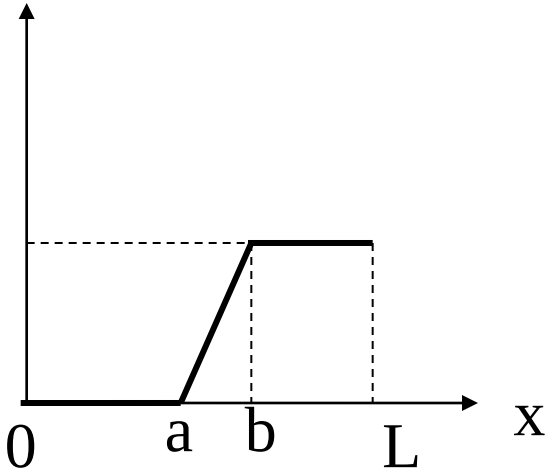


$$a = 50, b = 150, \alpha = 0.2, \beta = 2, \gamma = 1, y_a = 30, y_b = 200$$

What about Other Transforms?

Contrast Clipping

$$y = \begin{cases} 0 & 0 \le x < a \\ \beta(x-a) & a \le x < b \\ \beta(b-a) & b \le x < L \end{cases}$$

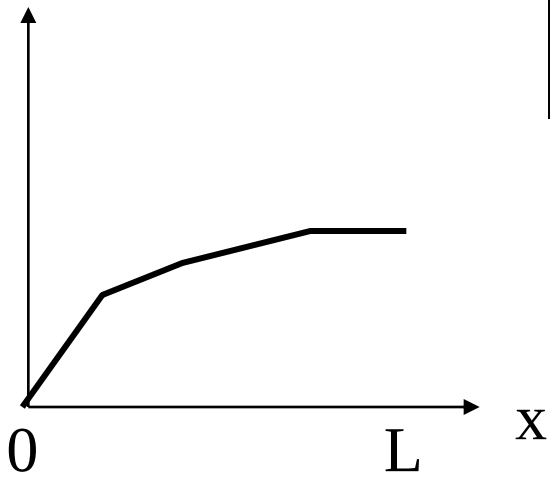


$a = 50, b = 150, \beta = 2$

What about Other Transforms?

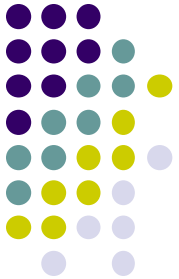
Range Compression

$$y = c \log_{10}(1 + x)$$



c = 100

Basic of Full Color Image Processing -1



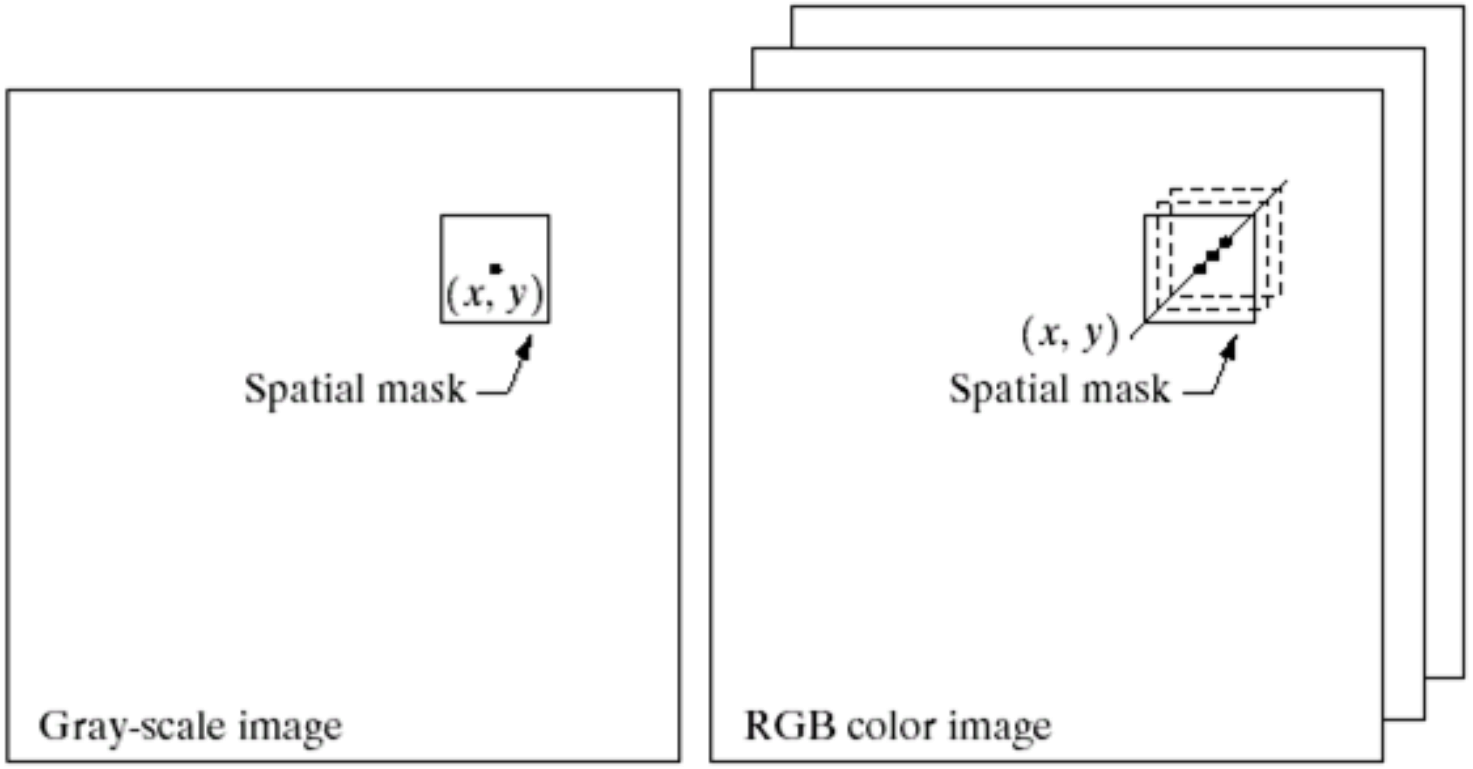
- Let c represent an arbitrary vector in RGB color space

$$c = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- For an image of size M*N,

$$c(x,y) = \begin{bmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix}$$

Basic of Full Color Image Processing -2



Simple Color Transformation

- Processing the components of a color image within the context of a color model.

$$g(x,y) = T\left[f(x,y)\right]$$

$$s_i = T_i(r_1, r_2, \ldots, r_n), \qquad i = 1,2,\ldots, n$$

Color components of ***g***

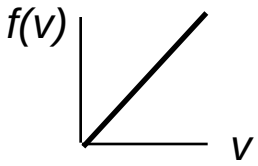
Color mapping functions

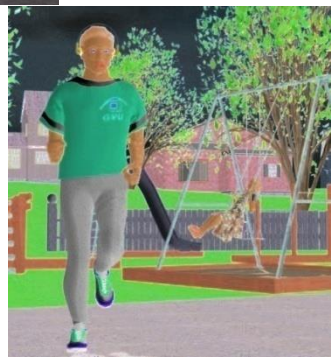
Color components of ***f***

Point Processing

- For color image, the effects are:

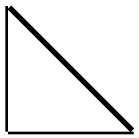
f(v) = v identity; no change

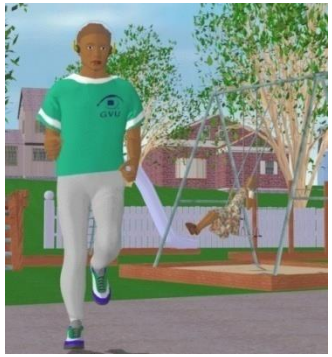




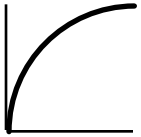
f(v) = 1-v negate an image

(black to white, white to black)

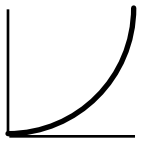




f(v) = v^p, p<1 brighten



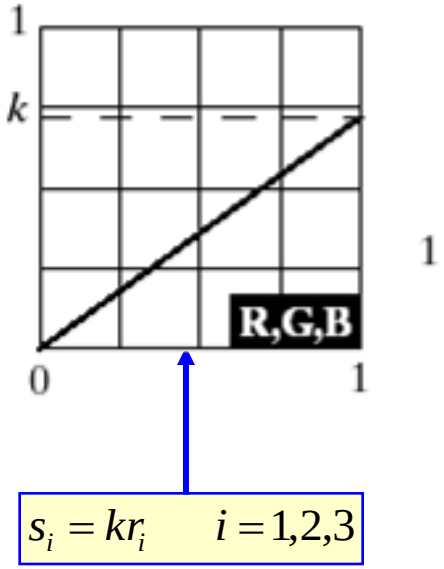
f(v) = v^p, p>1 darken



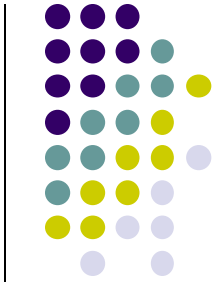
Modifying the Intensity

$$g(x, y) = kf(x, y)$$

(a) Original Image (b) Result of decreasing its intensity by 30%, i.e. let k = 0.7



Tone and Color Correction -1



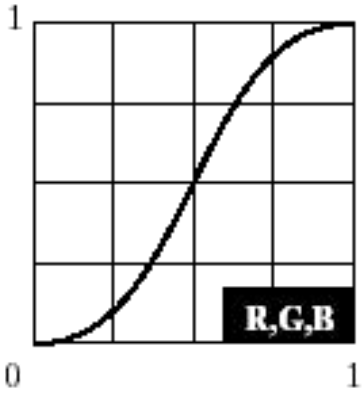
- The tonal range of an image, also called its key-type, refers to its general distribution of color intensities.
- High-key images: Most of the information is concentrated at high intensities.
- Low-key images: Most of the information is concentrated at low intensities.

Tone and Color Correction -2

- Mid key (or Mid-Tone) image: to highlight the light (high key) and dark (low key) range of image intensities

Flat

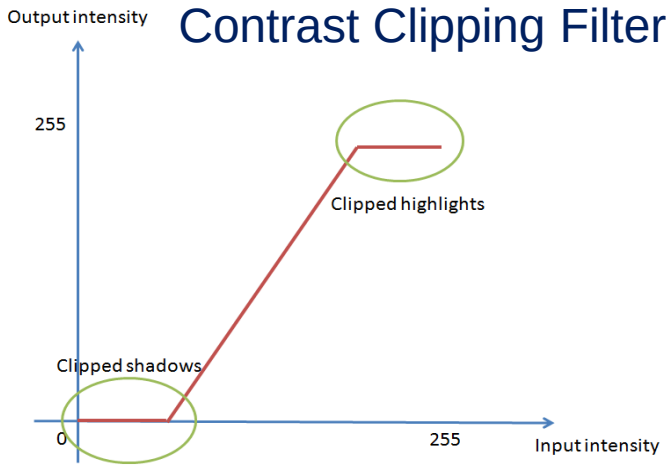
Corrected

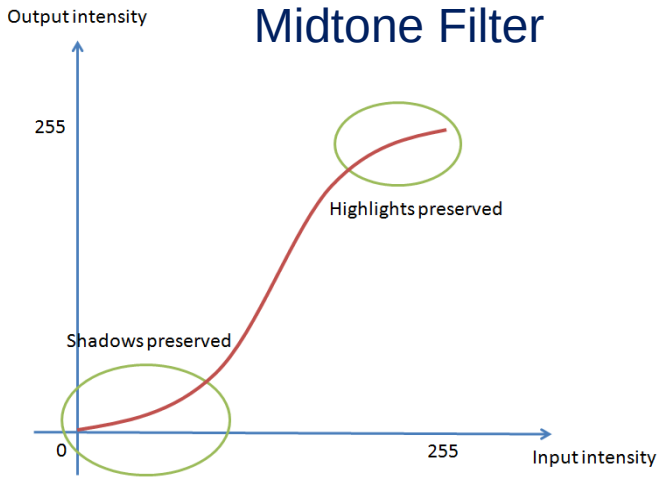


Note: By adjusting the RGB components equally does not alter the image hues

Tone and Color Correction -3

Mid-tone Filter





Increase contrast of midtones while preserving highlight/shadows

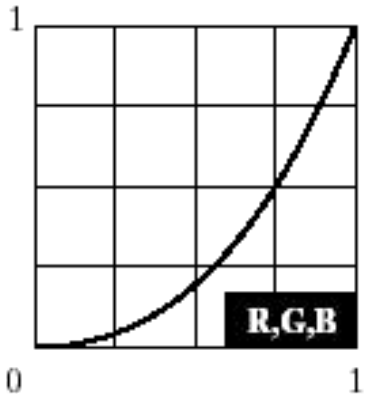
Image after midtone filter

Tone and Color Correction -4

- High key image: to highlight lower intensity range of image intensities

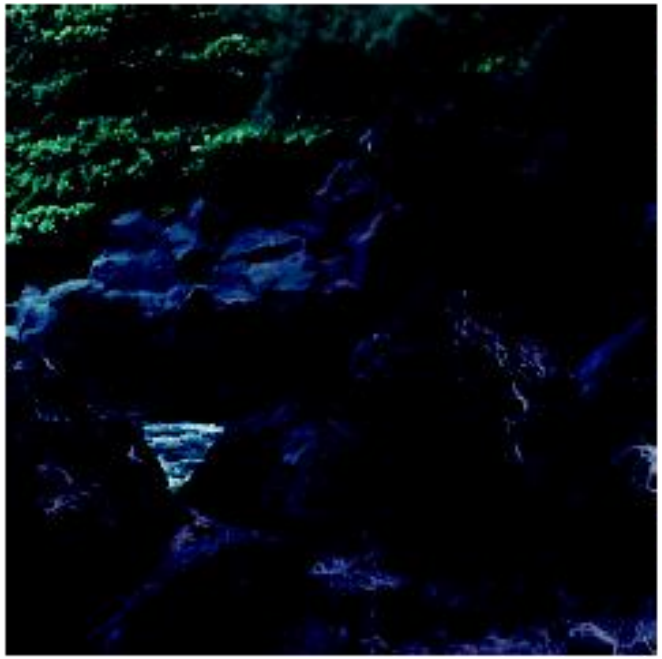
Light

Corrected



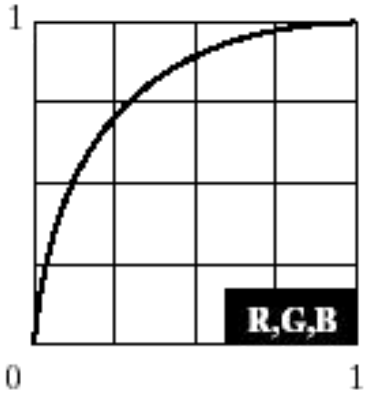
Tone and Color Correction -5

- Low key image: to highlight high intensity range of image intensities



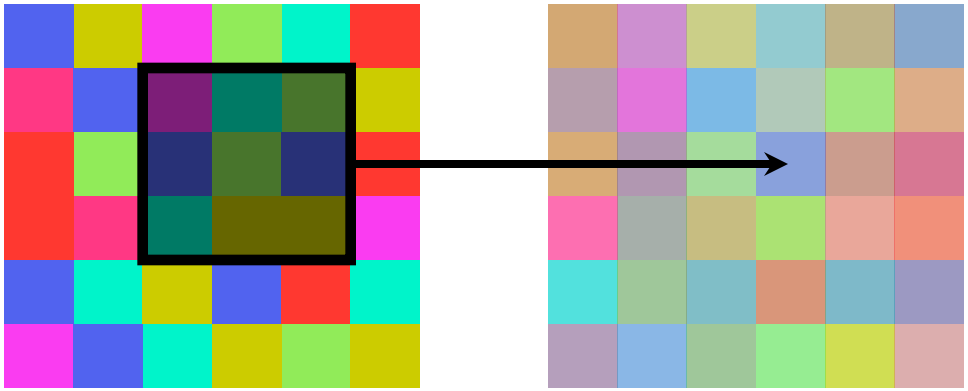
Dark

Corrected



Signals and Filtering

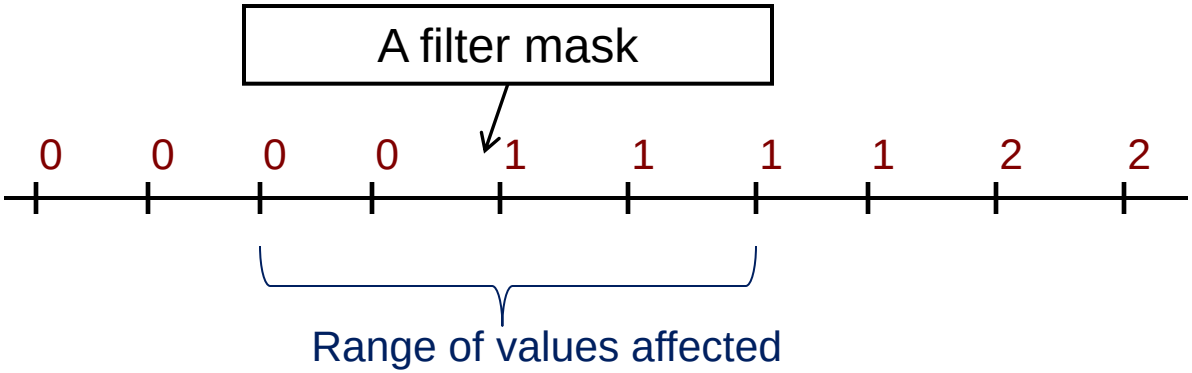
- Image is a 2D signal: color(x,y)
- Signals can be continuous or discrete
- Raster images are discrete
  - In space: sampled in x, y
  - In color: quantized in value
- Filtering: a mapping from signal to signal
- To introduce the concept of convolution



Convolution

- Used for filtering, sampling and reconstruction
- Example of Convolution in 1D
  - Apply a range filter across the 1D signals

    The mask covers 5 values in range

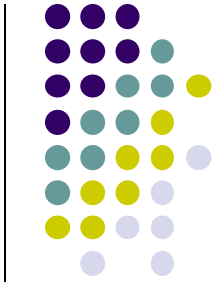


  - If the mask is [0.5, 0.5, 0.5, 0.5, 0.5], and the effect is summation, then apply mask to each point in range above will give:

    [0.5, 1.0, 1.5, 2.0, 3.0, 3.5, ..]

  - What if the result is to take medium value instead of summation?

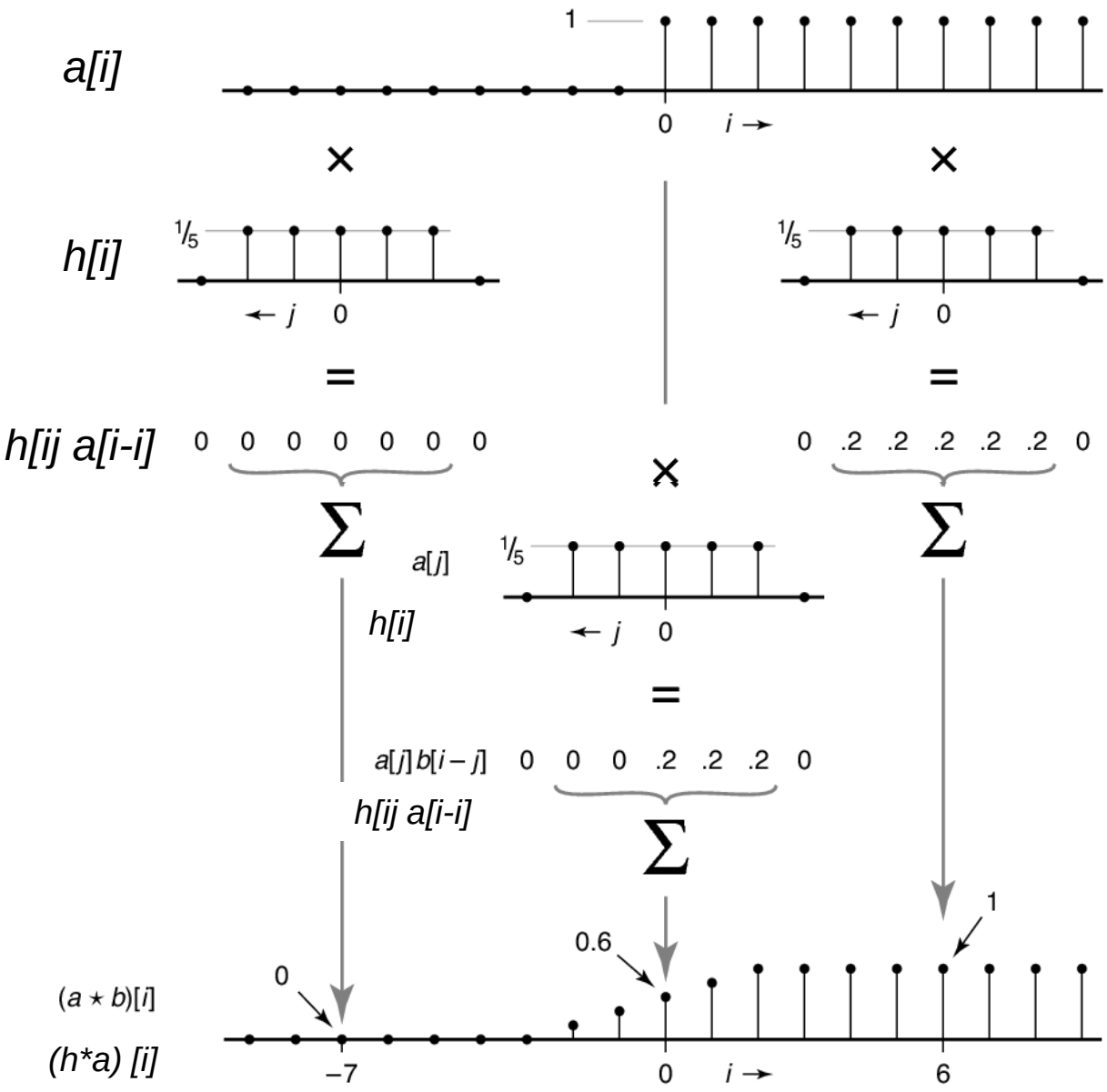
Convolution filters



- Mathematics of convolution in 1D
  - a(t) is input signal
  - b(s) is output signal
  - h(u) is filter

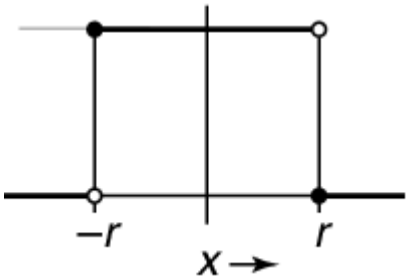
$$b(s) = \sum_{t=-\infty}^{+\infty} a(t) * h(s-t)$$

Convolve Example:

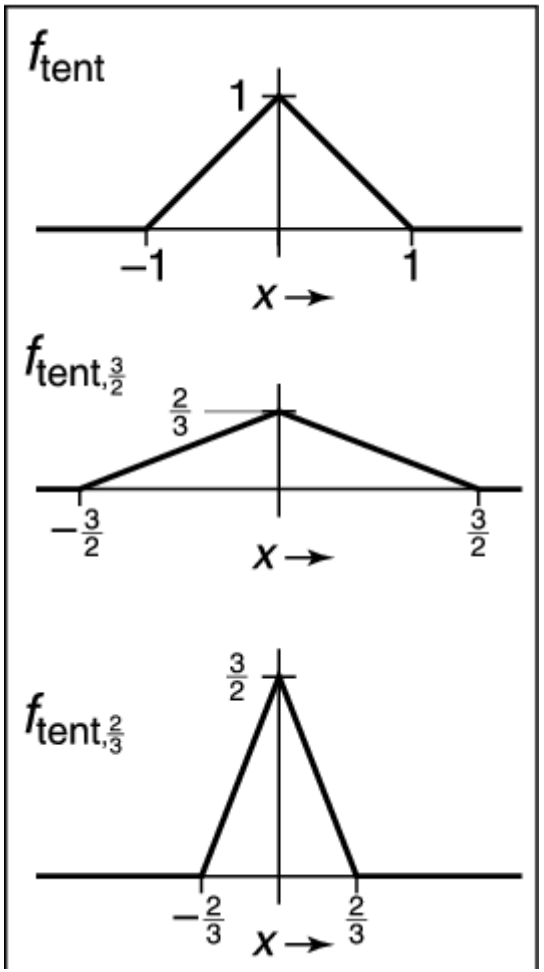


Convolution filters

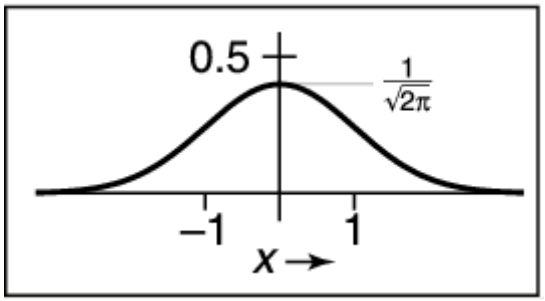
- Other forms of convolution filters



box

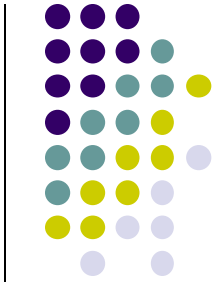


tent



gaussian

Convolution filters



- Mathematics of convolution in 1D
  - a(t) is input signal
  - b(s) is output signal
  - h(u) is filter

$$b(s) = \sum_{t=-\infty}^{+\infty} a(t) * h(s-t)$$

- Convolution in 2D

$$b(x,y) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} a(u,v) * h(x-u, y-v)$$

Filters with Finite Support

- Filter h(u,v) is 0 except in given region
- Represent h in form of a matrix
- Example: 3 x 3 blurring filter in matrix form:

$$h(u,v)=\begin{bmatrix} 1/9 & if\ -1\le u,v\le 1 \\ 0 & otherwise \end{bmatrix}$$

or

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Blurring Filters

- A simple blurring effect can be achieved with a 3x3 filter centered around a pixel,
- More blurring is achieved with a wider *n x n* filter:

Original Image

Blur 3x3 mask

Blur 7x7 mask

Image Filtering: Blurring

original, 64x64 pixels

3x3 blur

5x5 blur

- Blur Filtering:
  - Average values of surrounding pixels
  - Can be used for anti-aliasing
  - What do we do at the edges and corners?
- For noise reduction, use median, not average
  - Eliminates intensity spikes
  - Non-linear filter

Example: Noise Reduction -1

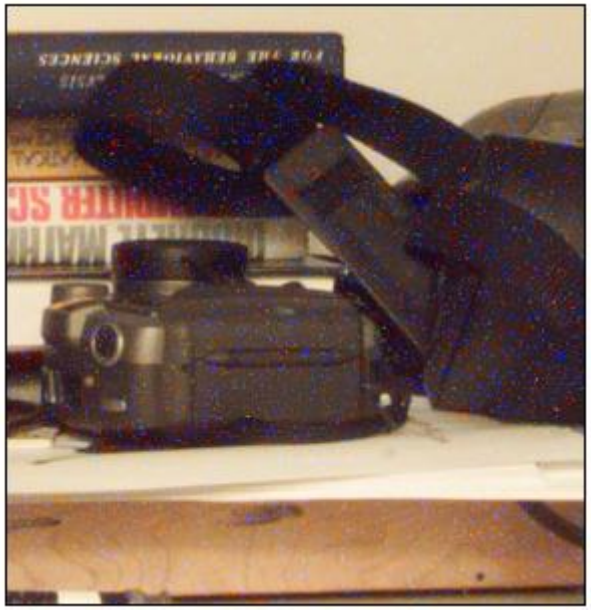


Image with noise

Median filter (5x5)

- A Demo of image filtering: http://joelb.me/jsmanipulate/

Example: Noise Reduction -2

Tom Ridge left the Pennsylvania governorship last October, when U.S. President George W. Bush appointed him to head the newly created Office of Homeland Security.

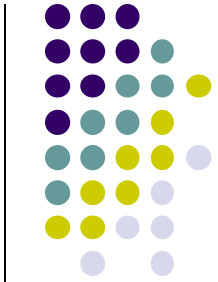
Original image

Tom Ridge left the Pennsylvania governorship last October, when U.S. President George W. Bush appointed him to head the newly created Office of Homeland Security.

Image with noise

Median filter (5x5)

Sobel Filter



- Edge detection filter, with some smoothing
- Approximate

$$\frac{\partial}{\partial x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \frac{\partial}{\partial y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Sobel filter is non-linear
  - Square and square root (more exact computation)
  - Absolute value (faster computation)

Sample Filter Computation

- Part of Sobel filter, detects vertical edges

-1	0	1
-2	0	2
-1	0	1

h

0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	25	25	25	25	25

a

0	0	0	0	25	25	0	0	0	0
0	0	0	0	25	25	0	0	0	0
0	0	[redacted]	0	25	25	0	0	0	0
0	0	0	0	25	25	0	0	[redacted]	0
0	0	0	0	25	25	0	0	0	0
0	0	0	0	25	25	0	0	0	0
0	0	0	0	[redacted]	25	0	0	0	0
0	0	0	0	25	25	0	0	0	0
0	0	0	0	25	25	0	0	0	0
0	0	0	0	25	25	0	0	0	0

b

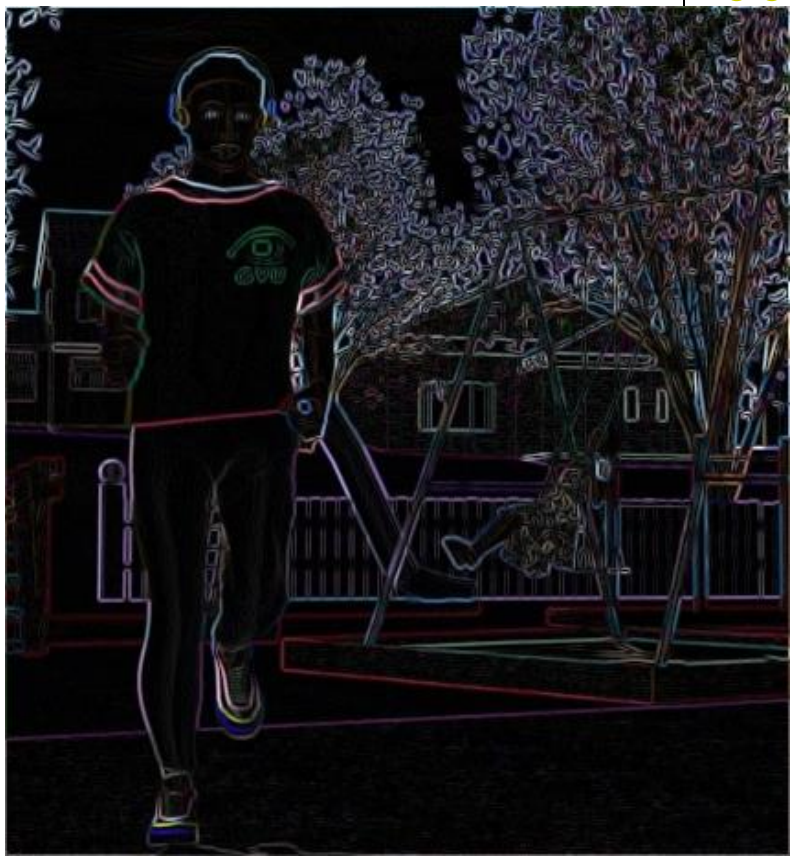
Example of Edge Filter

Tom Ridge left the Pennsylvania governorship last October, when U.S. President George W. Bush appointed him to head the newly created Office of Homeland Security.

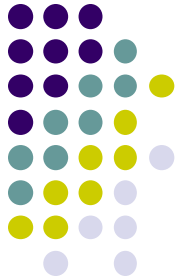
Original image

Edge filter, then brightened

Image Filtering: Edge Detection



Outline of Lecture

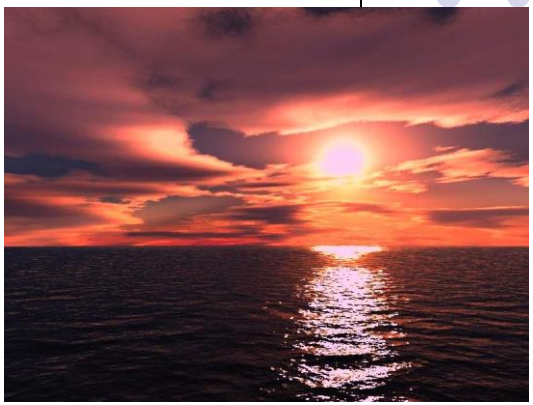


- Dithering
- Image Filters
- Color Transfer between Images
- Image Filtering App

Color Transfer - Introduction

Target image
(or Style image)

Source mage



Result image

Extract

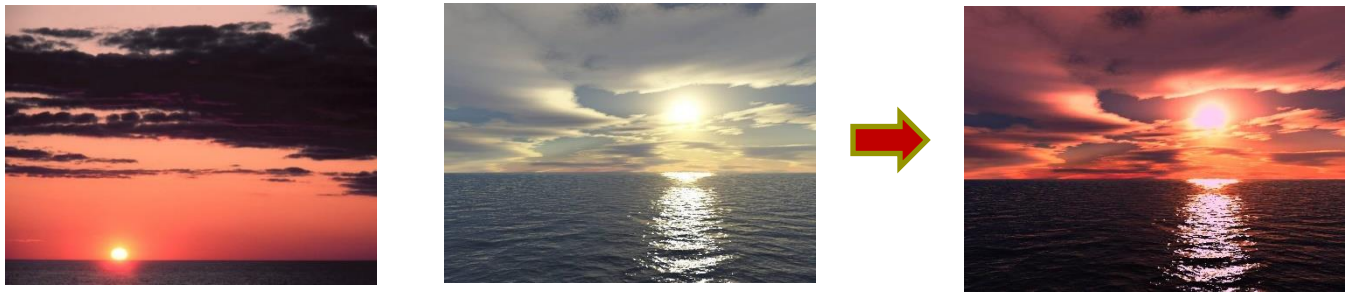
Transfer

Color Feature

Reinhard et al., 2001

Methods of Color Transfer

- Transfer style of target image to source image
- Global color transfer
  - This technique transfer color between the whole images
- Local color transfer
  - The images are segmented into local areas. Then, the transfer process is performed between local areas

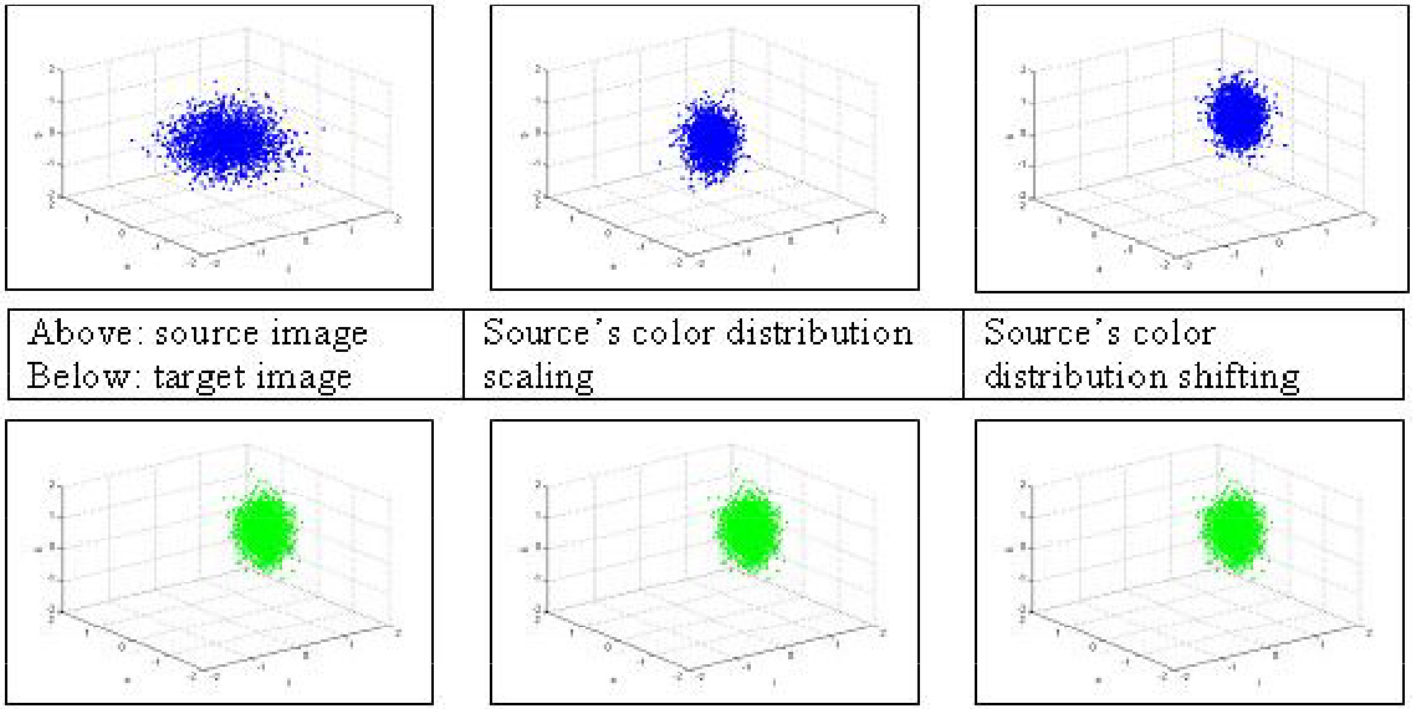


Target (Style) image Source image Result image

- Involves linear color transfer: scale and shift
  - Transfer to new color space
  - Compute mean and standard deviation along each color axis
  - Shift and scale style image to have same statistics as the source image

Global Color Transfer -1
Linear Transform

- Scale and shift the color distributions using *mean* & *std*.



Above: source image Below: target image	Source's color distribution scaling	Source's color distribution shifting

$$C' = \frac{\sigma_t}{\sigma_s}(C_s - \mu_s) + \mu_t$$

Global Color Transfer -2
Linear Transform

$$C' = \frac{\sigma_t}{\sigma_s}(C_s - \mu_s) + \mu_t$$

Where C' = new color

C_s = old color

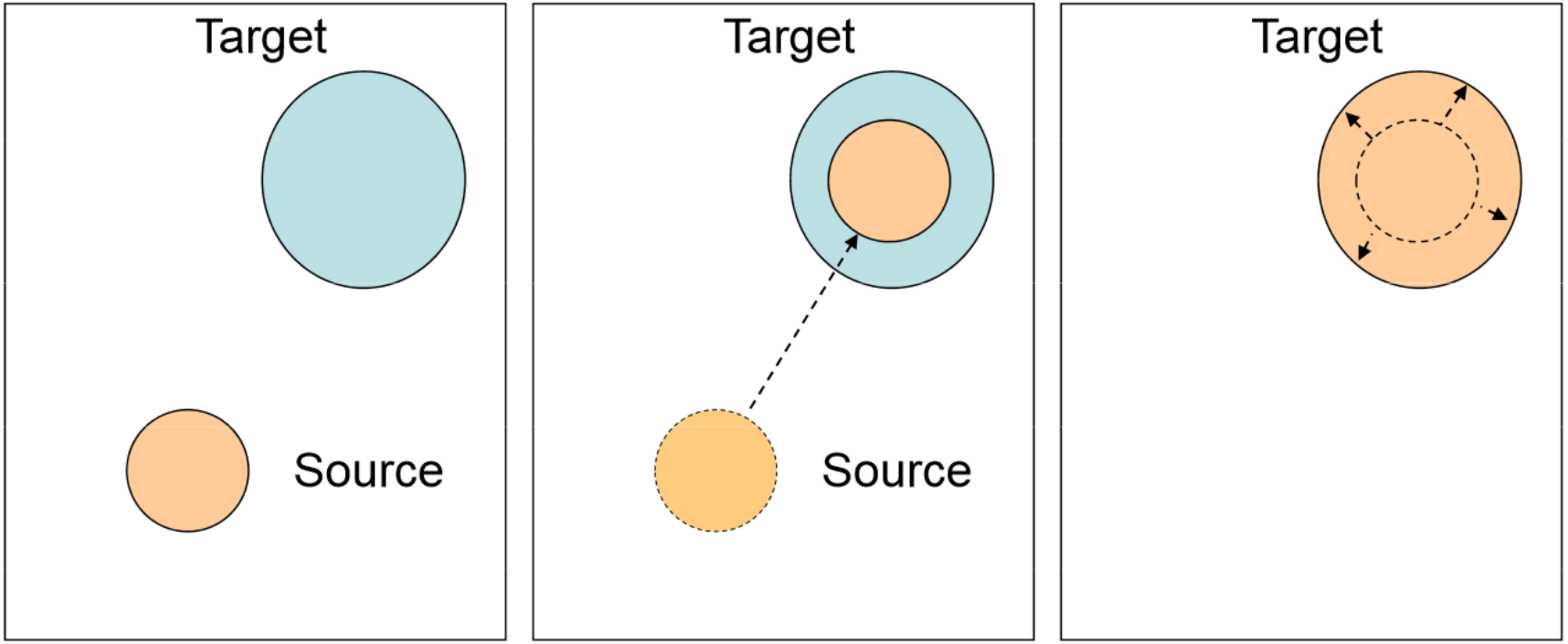
σ_t = SD of target image

σ_s = SD of source image

μ_t = mean of target image

μ_s = mean of source image

Global Color Transfer -3
Linear Transform



- Perform shifting of source to target using means μ
- Follow by scaling of source to match the target using σ
- Must be done in appropriate color space

Why not use RGB space?

- In RGB, there are strong correlations between axis
- Need to use other color space, such as Lαβ (see notes)

Input images

Output images

RGB

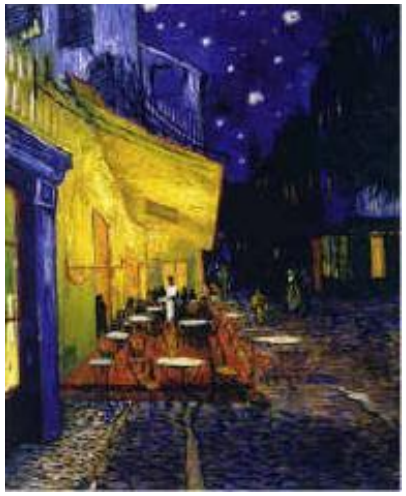
Lαβ

Color Transfer Examples

Target image

Source image

Result image



More Color Transfer Examples

Style image

Source image

Result image

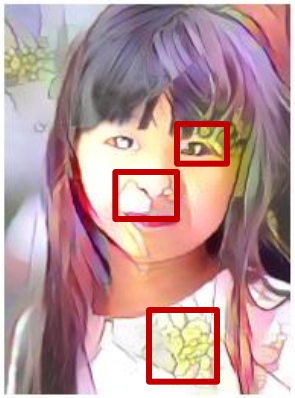
Color Transfer Summary

- Changing the statistics along each axis independently allows one image to resemble a second image
- If the composition of the images is very unequal, an approach using small swatches may be used successfully
- It works well on natural images, but there will be artifacts for object with clear shapes?

What Are the Problems?

(a) Content image

(b) Style image



(c) Gatys-style

(c) Lapstyle output

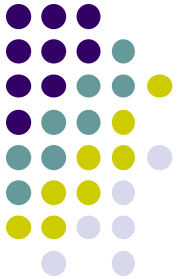
(d) Content Laplacian

(e) Gatys-style Laplacian

(f) Lapstyle Laplacian

- There are artifacts on the face.. WHY??
- Needs method to preserve the detailed structures
- Methods that preserve Laplician of image:
  (Laplacian-Steered Neural Style Transfer, ACMMM 2017)

Outline of Lecture



- Dithering
- Image Filters
- Color Transfer between Images
- Image Filtering App

Filter in Instagram

Features:

- Provides a fast and fun way to share your life with friends through a series of pictures
- “Snap a photo with your mobile phone, choose a filter to transform the look and feel, send to FB, Twitter or Flickr – it’s all as easy as pie. It’s photo sharing, reinvented”, from Instagram Web site
- Commonly adopted in all mobile image systems and mobile phones

Most Popular Instagram Filters

http://www.huffingtonpost.com/2014/04/15/instagram-filters-ranked_n_5127006.html

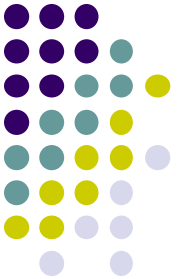
1) Early Bird

2) Mayfair

3) Amaro

4) Rise

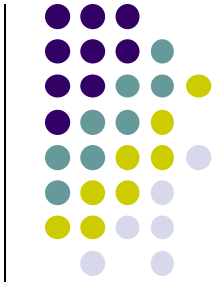
Programming Assignment 2



A program that offers a set of Instagram-like filters to transform an image such as:

- Negate an image
- Change brightness of the image
- Change darkness of the image
- Style transfer from the image

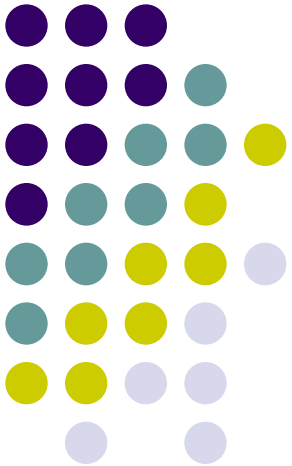
Summary



- Discussed various approaches to process images
  - Dithering
  - Filtering
  - Form basis for Image Filtering Applications
- Next lesson:
  - Compression Algorithm

Introduction to Media Computing

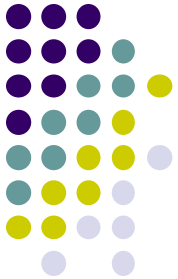
Lecture 11: Multimedia Compression Techniques



Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

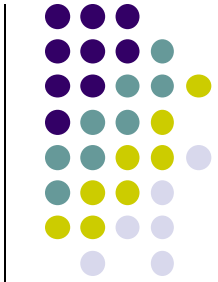
TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

IMC Topics



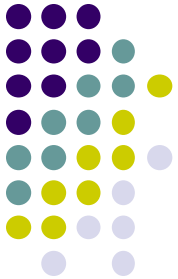
- Introduction to Media Computing
- Introduction to Text Retrieval
- Image Representation and Retrieval
- Multimedia Indexing and Search Architecture
- Fundamentals of Digital Image
- Introduction to Digital Audio
- Image Filters &Transformation
- **Multimedia Compression Techniques**
- The Lossy Compression Techniques, Image Standards: JPEG
- Color Model and Color JPEG
- Digital Video and MPEG Standards
- H261 & Other Video Standards
- Image/Video Applications and Future Trends

Outline of Lecture



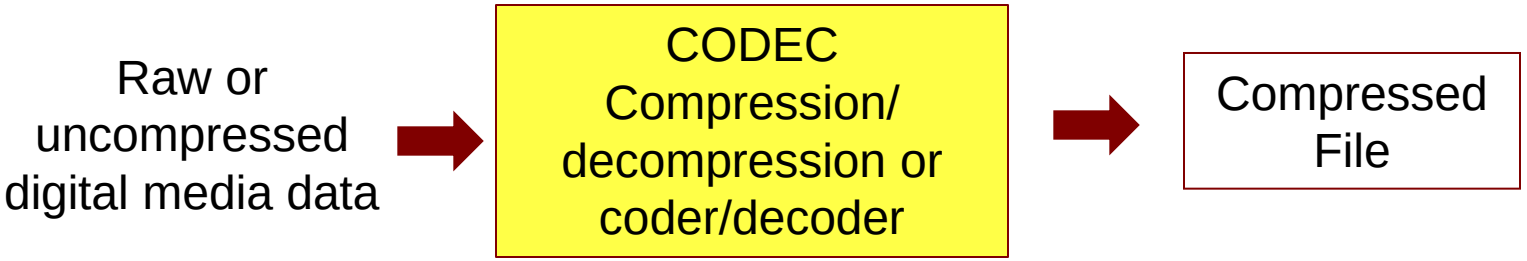
- Basic Digital Compression
- Huffman & Arithmetic Coding
- LZW Coding
- Predictive Encoding Scheme
- REFERENCES:
  - Z.N. Li and M.S. Drew (2003). Fundamentals of Multimedia. Prentice Hall (Chapter 7)

Digital Compression Introduction



- Media files require a huge amount of disk space
- Compression is needed for 2 reasons:
  - To reduce storage requirements
  - Most importantly, to reduce data transfer time.

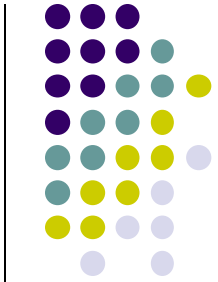
    Example, to play back an audio file, the disk and processor need to process about 1.4 million bps of data without compression



- Basic principle of compression:
  - Find and reduce redundant information within media file
  - But then what is the definition of redundancy?

    At code level or at perception levels

Types of Compression



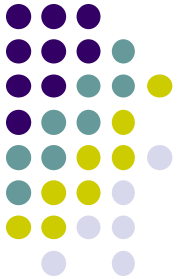
- Compression ratio

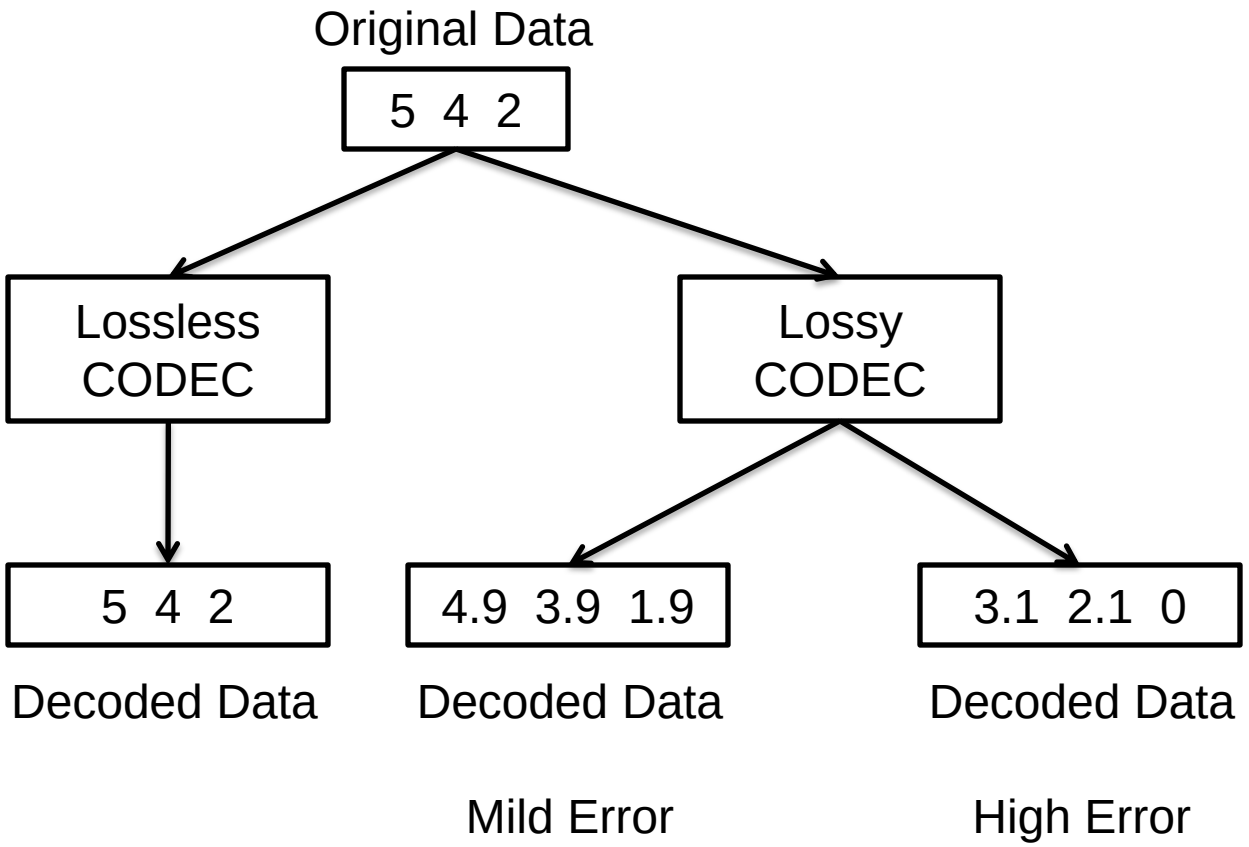
$$compression\,ratio = \frac{B_0}{B_1}$$

- Lossless compression:
  - Attempts to represent existing info in a more compact form without discarding any data
  - Pros: the original data stays intact and can be reused
  - Cons: compression achieved is not high: 2 to 5 times

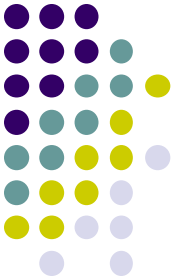
- Lossy compression
  - Parts of the original data are discarded permanently
  - Original data not recoverable → degradation of media quality
  - Balance: compression ratio vs. human perception quality
  - High compression ratios: > 10 to 100 times

Lossless vs. Lossy Compression





Lossless vs. Lossy Compression



Original Picture 800 x 600

1.37 MB

Decoded Picture (Low Losses)

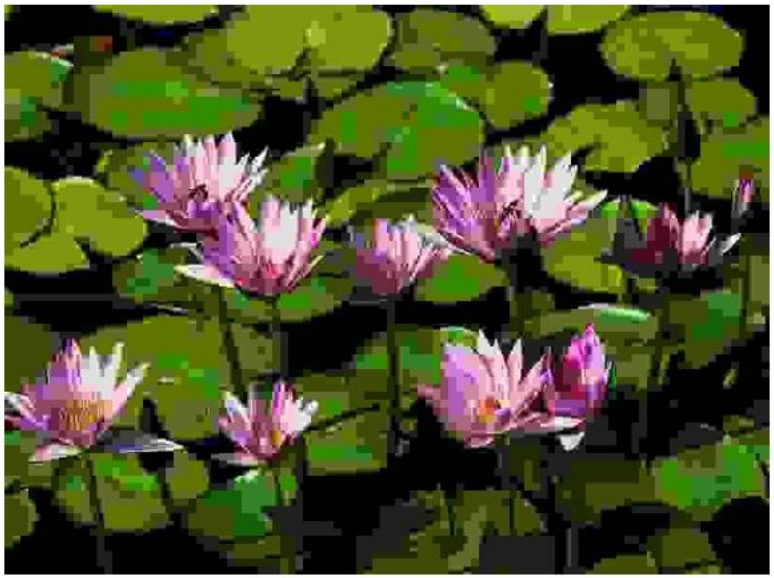
85 KB

Lossless vs. Lossy Compression

Original Picture 800 x 600

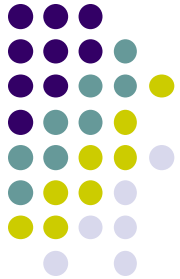
1.37 MB

Decoded Picture (High Losses)



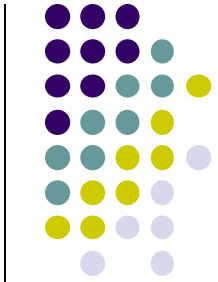
16 KB

Key Idea in Lossy Compression:
Exploit Redundancy



- Statistical redundancy:
  - Exploit statistical relationships existing within the media data
  - Example: Sequence of pixels with similarity intensity values in an image
- Psycho-visual redundancy:
  - Visual/ auditory information is not perceived equally, some are more important than other information
  - We can use fewer data to represent less important visual information without affecting human perception
  - This is most important for multimedia contents!!

Basic Compression Techniques



3 classes of techniques

ENTROPY

- Uses knowledge of image contents to reduce coding redundancy
- Ignore media and human visual characteristics
- Example: Run-length Coding, Huffman Coding

SOURCE

- Takes into account of semantics or characteristics of data
- Certain info has less relative importance than others in normal visual processing
- Use quantization - i.e. quantize to coarser values for less important info

HYBRID

- Most standards use diff schemes at different stages of compression.

Information Theory

- Given an info source with n symbols S = [s_1, s_2, … s_N] and corresponding probabilities P = [p_1, p_2, … p_N], its entropy Λ is defined as:

$$\eta = H(S) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{N} p_i \log_2 p_i$$

  - η is called entropy – a measure of disorder of the system
  - $\log_2(1/p_i)$ is called self info, or # of bits needed to encode s_i

    Thus, the higher the occurrence probability of a symbol, the lower is the number of bits needed for its encoding
  - |η| gives the min average # of bits needed to encode the symbols
  - Shannon coding theory states that the entropy is the best we can do to encode the set of symbols:

    i.e. $\eta \leq \bar{l}$, where $\bar{l}$ is the actual # of bits used

Information Theory -2

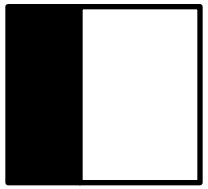
- Given a monochrome image with 2 symbols [s_1, s_2]
- If 2 symbols are uniformly distributed, then

$$|\eta| = \sum_{i=1}^{N} p_i \log_2 p_i = \frac{1}{2}\log(2) + \frac{1}{2}\log(2) = 1$$

➔ 1 bit is needed to encode each symbol on average

- If occurrence probability of s_1 = 1/3 and s_2 = 2/3, then:

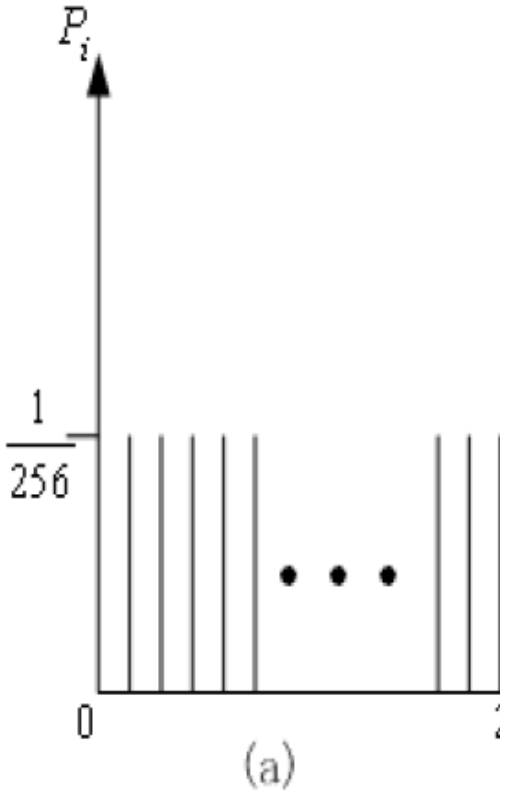
$$|\eta| = \sum_{i=1}^{N} p_i \log_2 p_i = \frac{1}{3}\log(3) + \frac{2}{3}\log(\frac{3}{2}) = 0.92$$

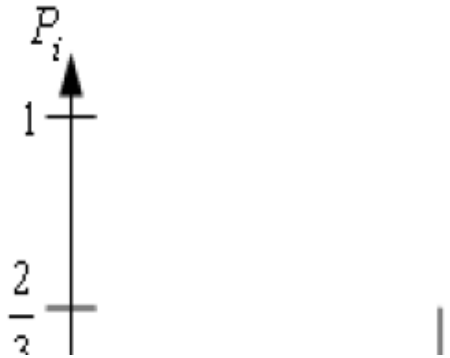


➔Only 0.92 bit is needed on average

➔Actual # of bits used >= |η|

- Key Finding:
  - o When probability distribution is skewed, compression is possible
- The result can be extended to a group of pixels





The entropy of the two above images

- The entropy of image a is:
-
- $$\eta = \sum_{i=0}^{255} \frac{1}{256} \cdot \log_2^{256} = 8$$

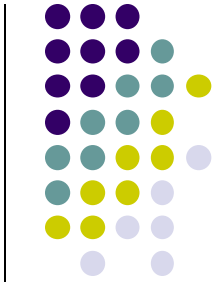
- The entropy of image b is:

$$\eta = \frac{1}{3} \cdot \log_{\frac{3}{2}} + \frac{2}{3} \cdot \log_{\frac{3}{2}}^{\frac{3}{2}}$$

$$= 0.33 \times 1.59 + 0.67 \times 0.59 = 0.92$$

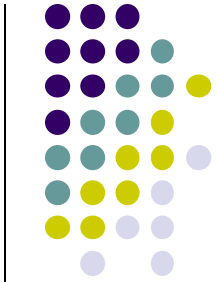
Histograms fo

Run-Length Coding (RLC)



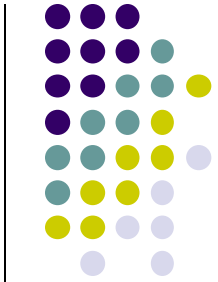
- RLC replaces repeated byte sequences with the byte and the number of occurrences.
- The number of occurrences is indicated by a special flag like "!"
  E.g. : BBBBBBBBB -->B!9
- Uncompressed sequence:
  ABCCCCCCCCCDEFFFFGGG (length 20 bytes)
- Compressed sequence:
  ABC!9DEF!4GGG (length 13 bytes)
- Effective if there are sequences of 4 or more characters
- It is easy to implement; and useful for text files or images which have large regions of uniform colors (or gray values)
- It is used in TIFF and PDF files

Outline



- Basic Digital Compression
- Huffman & Arithmetic Coding
- LZW Coding
- Predictive Encoding Scheme

Huffman Coding -1



- Given a set of symbols, the easiest approach is to do a *fixed-length coding*
- For example, if the symbols of a source are a_1, a_2,… , a_8, then the natural code or the gray code could be used:

  $a_1 = 000$ $a_2 = 001$ $a_3 = 010$ $a_4 = 011$

  $a_5 = 100$ $a_6 = 101$ $a_7 = 110$ $a_8 = 111$

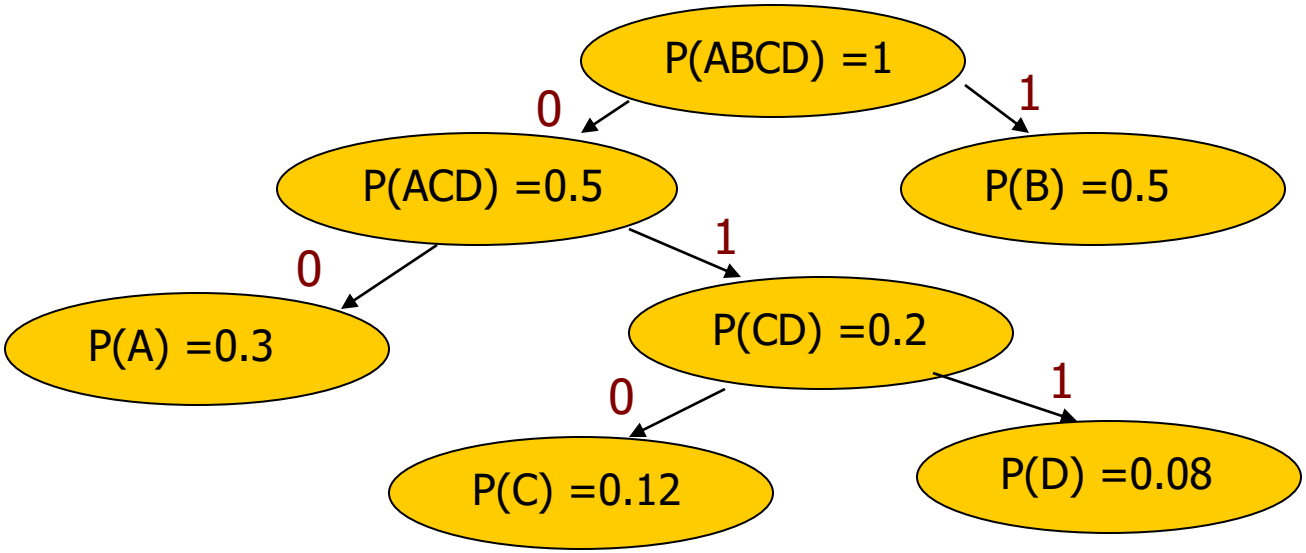
  The number of bits per code = 3 → No compression

- Huffman coding aims to derive more optimal codes by utlizing occurrence probabilities of codes
- It exploits information theory by assigning each symbol with a different integer bit length according to its $\log_2(1/p_i)$

Huffman Coding -2

- Consider a source with the following symbol occurrence

Symbol	A	B	C	D
Probability	0.3	0.5	0.12	0.08



$\bar{l}$, Avg length of this code =

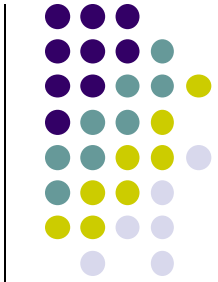
0.3*2 + 0.5*1 + 0.12*3 + 0.08*3

= 1.7 bits/symbol

or $\bar{l} \geq \eta$

- Code(A) =00, Code(B)=1, Code(C)=010, Code(D) = 011

Huffman Coding -3



- Idea: more frequent symbols -> shorter codewords
- Algorithm (a bottom up approach):

Step 1: Initialization: put all symbols on the list sorted according to their frequency counts

Step 2: Repeat until the list has only one symbol left

(a) From the list, pick two symbols with the lowest frequency counts, and form a Huffman sub-tree

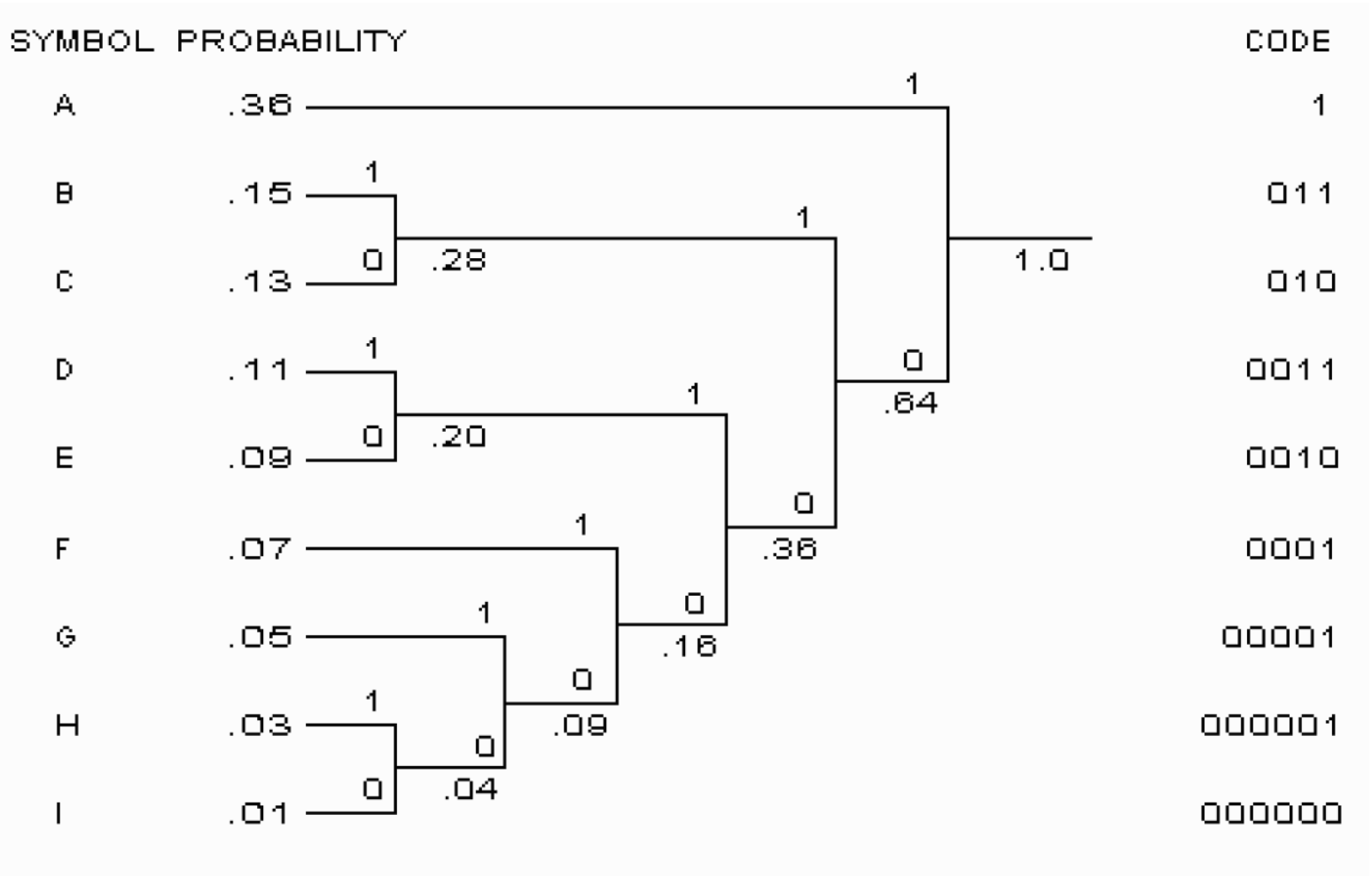
(b) Assign the sum of the children's frequency counts to the parent and insert it into the list

(c) Delete the children from the list

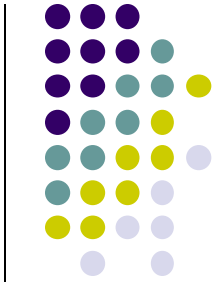
Step 3: Assign a codeword for each leaf based on the path from the root

Huffman Coding -4

- More Example



Huffman Coding -5



Properties of Huffman Codes

- Unique prefix property:

  No Huffman code is a prefix of any other Huffman codes

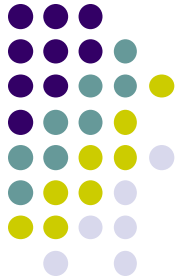
- Optimality:

  It is a min. redundancy code

  It can be shown that: $\eta \leq \bar{l} \leq \eta + 1$

- Can be applied to one symbol at a time (scalar coding), or a group of symbols (vector coding), or one symbol conditioned on previous symbols (conditional coding)

Arithmetic Coding -1



- Problem with Huffman Coding:
  - It requires at least 1 bit per symbol
  - Consider a message with probability 0.999; its self info is:
    -log(.999) = 0.00144
  - Theoretically sending 1,000 such message requires only 1.44 bits, but using Huffman codes, we require 1,000 bits
- Solution:
  - Allowing blending of bits in a message sequence
    Hence may only requires 3 bits for the example
    ➔ Arithmetic Coding
  - Use in PPM, JPEG/MPEG (as option), DMM
  - More expensive than Huffman coding, but integer implementation is not too bad

Arithmetic Coding -2

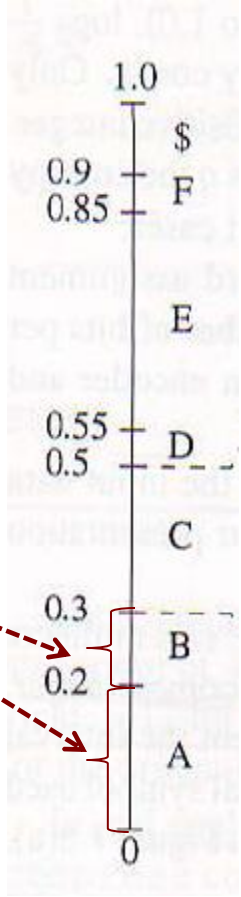
- Basic idea of solution:
  - Encode message at a time, rather than symbols
- Approach
  - Represent a sequence of symbols by an interval with length equal to its probability
  - The interval is specified by its lower boundary (*l*), upper boundary (*u*) and length *d* (=probability)
  - The codeword for the sequence is the common bits in binary representations of *l* and *u*
  - *A more likely sequence=a longer interval=fewer bits*

  - A message is represented by a half-open interval [*a*, *b*) where *a* and *b* are real numbers between 0 and 1. Initially, the interval is [0, 1). When the message becomes longer, the length of the interval shortens and the number of bits needed to represent the interval increases

Arithmetic Coding -3

- EXAMPLE:
  - Probability distributions of 7 symbols
  - The intervals (probabilities) of the 7 symbols

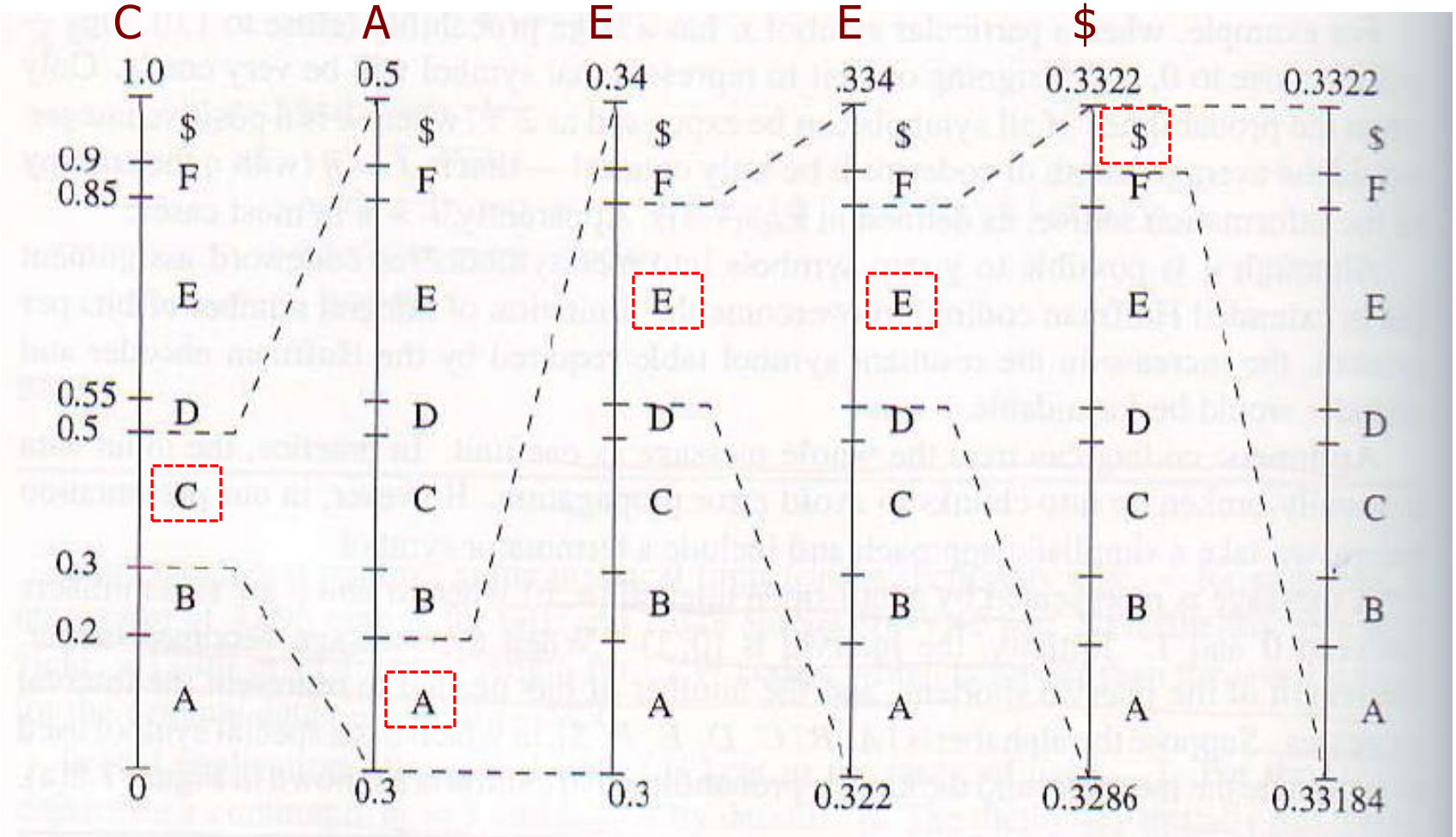
Symbol	Probability	Range
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
$	0.1	[0.9, 1.0)



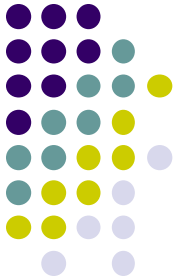
Arithmetic Coding -4

- EXAMPLE:
  - Given a message sequence CAEE$
  - Arithmetic Coding process

Symbol	Probability	Range
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
$	0.1	[0.9, 1.0)



Arithmetic Coding -5
The Algorithm



```
BEGIN
   low = 0.0;     high = 1.0;       range = 1.0;

   while (symbol != terminator)
   {
       get (symbol);
       low = low + range * Range_low(symbol);
       high = low + range * Range_high(symbol);
       range = high - low;
   }

   output a code so that low <= code < high;
END
```

- Not a pre-fix code

Arithmetic Coding -6

- EXAMPLE:
  - New low-high, range generated

Symbol	low	high	range
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.30	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.33220	0.00036

Symbol	Probability	Range
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
$	0.1	[0.9, 1.0)

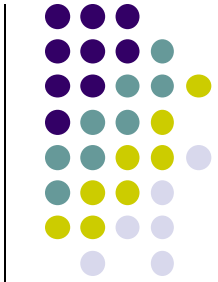
NOTE:

- Range = P_C x P_A x P_E x P_E x $P_\$$ = 0.00036
- Code for this message is between low & high: Low <= Code < High
- The binary codeword is: 0.0101010101 (= 0.33203123)

- Algorithm for generating Codeword for Encoder:

```
BEGIN
Code=0; k=1;
while (value(code) < Low) {
 assign 1 to the kth binary fraction bit;
 if (value(code) > high) replace the kth code by 0;
 k = k+1;
} END
```

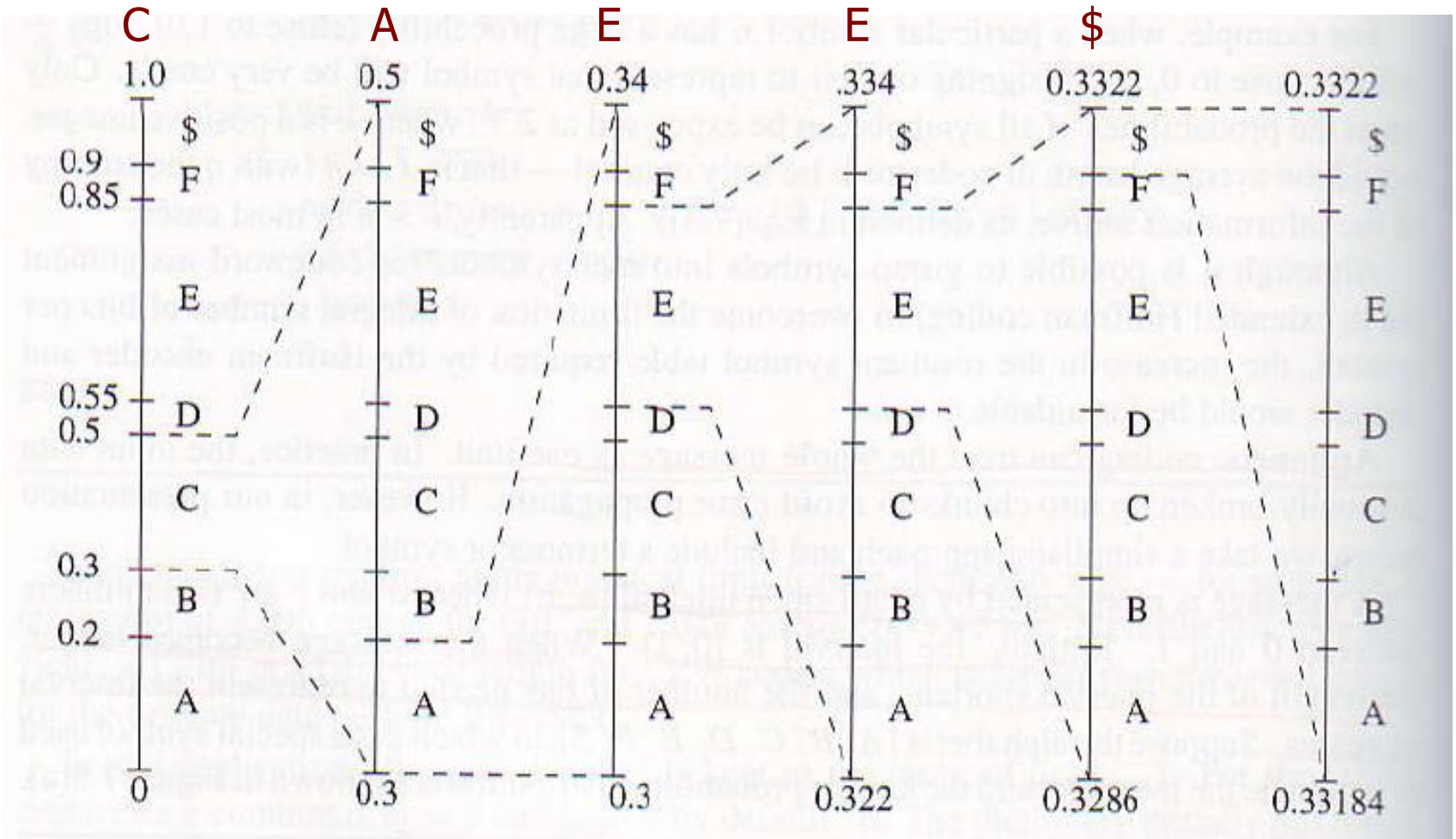
Arithmetic Coding -7
The Decoder



```
BEGIN
   get binary code and convert to
   decimal value = value(code);
   Do
   {
       find a symbol s so that
              Range_low(s) <= value < Range_high(s);
       output s;
       low = Rang_low(s);
       high = Range_high(s);
       range = high - low;
       value = [value - low] / range;
   }
   Until symbol s is a terminator
END
```

Arithmetic Coding -8

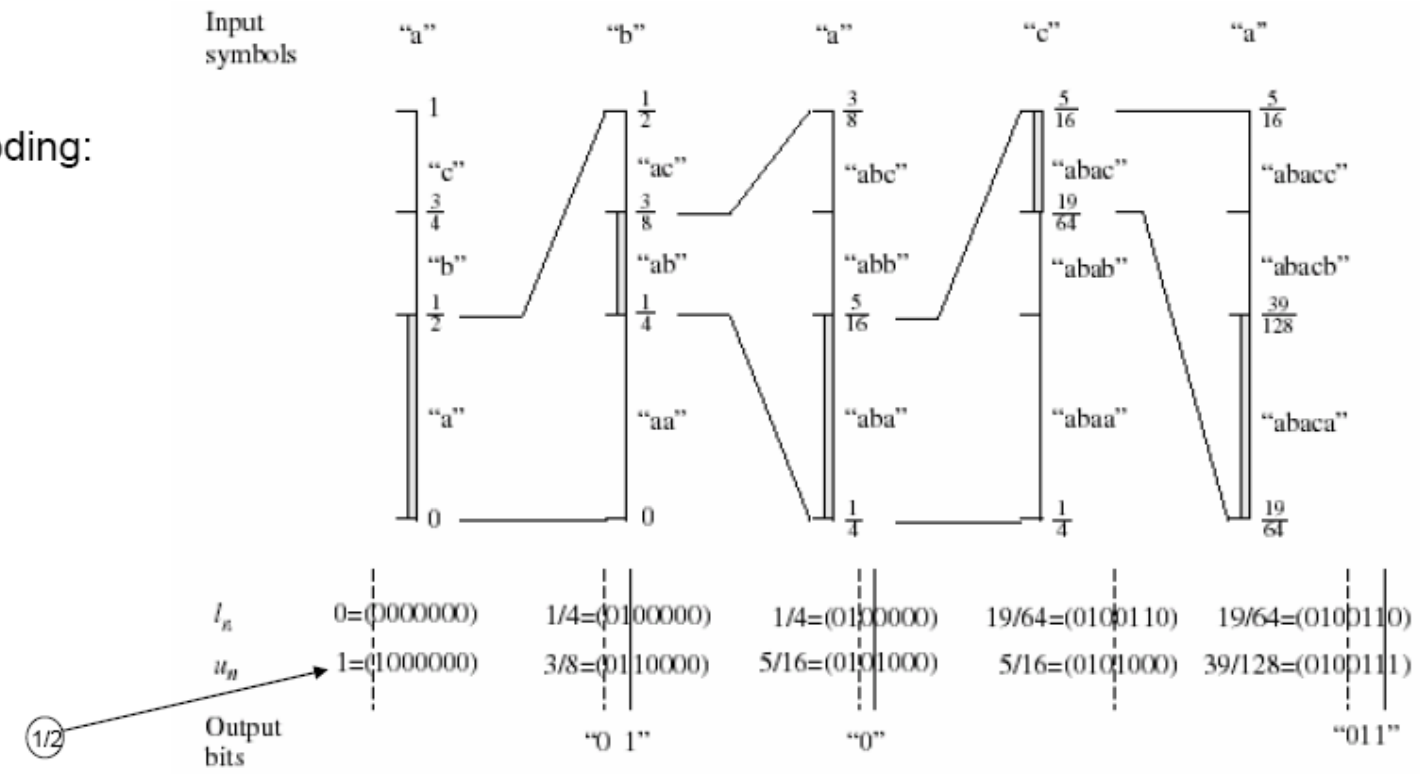
- Decoding:
  - Given the code 0.0101010101 or 0.33203123 in decimal
  - Decoding is simply the inverse of coding process



Arithmetic Coding -9
Another example

P(a)=1/2
P(b)=1/4
P(c)=1/4

Encoding:



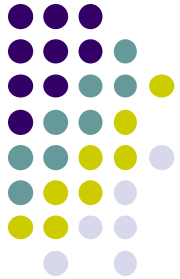
(a)

Decoding:

Received bits	Interval	Decoded symbol
"0"	[0,1/2)	"a"
"01"	[1/4,1/2)	—
"010"	[1/4,3/8)	"b"
"0100"	[1/4,5/16)	"a"
"01001"	[9/32,5/16)	—
"010011"	[19/64,5/16)	"c"
…	…	…

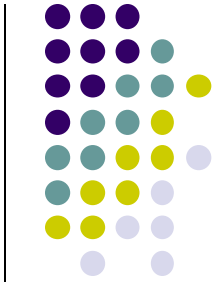
(b)

Outline



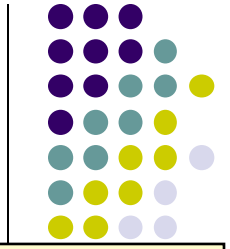
- Basic Digital Compression
- Huffman & Arithmetic Coding
- LZW Coding
- Predictive Encoding Scheme

LZW Algorithm -1
A Dictionary-based Coding Scheme



- Another approach: Build the dictionary adaptively
- It uses fixed-length codewords to represent variable-length strings of symbols/characters that commonly occur together, e.g., words in English text
- Its encoder and decoder build up the same dictionary dynamically while receiving the data
- LZW (Lempel-Ziv-Welch) places longer and longer repeated entries into a dictionary, and then emits the *code* for an element, rather than the string itself, if the element has already been placed in the dictionary.

LZW Algorithm -2
A Dictionary-based Coding Scheme



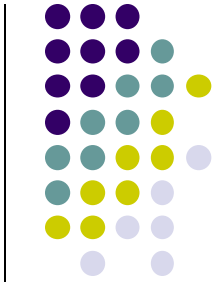
- LZW compression algorithm:
  - replaces strings of characters with single codes
  - It does not perform global analysis of text collection
  - Instead it analyses the incoming text on the fly, and add every new string found to table of string with a new code
  - Compression occurs when a single code is used (many times) instead of a string of characters
  - It is a lossless dictionary based compression algorithm

- The LZW Compression Algorithm is:

```
w = 1st input character;
while (read a character c)
{   if wc exists in the dictionary
        w = wc
    else {
        output the code for w
        add wc to the dictionary
        w = c
    }
}
```

- Original LZW uses dictionary with 12-bit codes (4096 entries)
  - 0-255 for ASCII codes
  - 250-4095 for new sub-strings

LZW Algorithm: An example



Let's start with a very simple dictionary (also referred to as a "string table"), initially containing only 3 characters, with codes as follows:

Code	String
1	A
2	B
3	C

Now if the input string is "ABABBABCABABBA", the LZW compression algorithm works as follows:

LZW Algorithm: An Example

Input string = "ABABBABCABABBA" – 14 symbol

w	c	Output	Code+ Symbol (wc)
A	B		
B	A		

The LZW Algorithm is:

```
w = 1st input character;
while (read a character c)
{   if wc exists in the dictionary
        w = wc
    else {
        output the code for w
        add wc to the dictionary
        w = c
    } }
```

S	C	Output	Code	String
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

LZW Algorithm: Coding Process

w	C	Output	Code	String
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

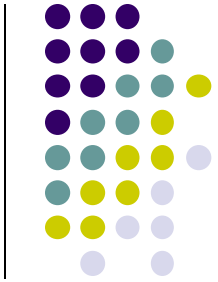
- The LZW Compression Algorithm is:

```
w = 1st input character;
while (read a character c)
{   if wc exists in the dictionary
        w = wc
    else {
        output the code for w
        add wc to the dictionary
        w = c
    }
}
```

Input is: ABABBABCABABBA (14 symbols of 8-bit each)

Output is: 124523461 (9 codes of > 8-bit each)

LZW Algorithm: Decoding Process



LZW decompression for string “ABABBABCABABBA”.

Input codes to the decoder are 1 2 4 5 2 3 4 6 1.
The initial string table is identical to what is used by the encoder.

S	K	Entry/output	Code	String
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

```
BEGIN
   s = NIL;
   while not EOF
   {
     k = next input code;
     entry = dictionary entry for k;
     output entry;
     if (s != NIL)
             add string s + entry[0] to dictionary with a new code;
     s = entry;
   }
END
```

LZW Algorithm –Another Example

Input string = "^ W E D ^ W E ^ W E E ^ W E B ^ W E T" – 19 symbols

w	c	Output	Code + Symbol (wc)
nil	^		
^	W		
W	E		
E	D		
D	^		
^	W		
^W	E		
E	^		
^	W		
^W	E		
^WE	E		
E	^		
E^	W		
W	E		
WE	B		
B	^		
^	W		
^W	E		
^WE	T		
T	EOF		

- The LZW Compression Algorithm is:

```
w = 1st input character;
while (read a character c)
{  if wc exists in the dictionary
       w = wc
   else {
       output the code for w
       add wc to the dictionary
       w = c
   }
}
```

LZW Algorithm –Another Example

Input string = "^ W E D ^ W E ^ W E E ^ W E B ^ W E T" – 19 symbols

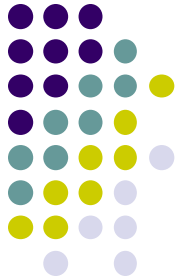
w	c	Output	Code	Symbol (wc)
nil	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

- The LZW Compression Algorithm is:

```
w = 1st input character;
while (read a character c)
{     if wc exists in the dictionary
            w = wc
      else {
            output the code for w
            add wc to the dictionary
            w = c
      }
}
```

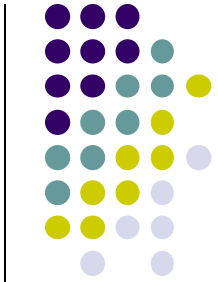
- Input: 19 symbols of 8-bit each
- Output: 12 codes of > 8-bit each
- Each code will need more than 8 bits, say 9 bits.
- Usually, compression doesn't start until a large number of bytes (e.g., > 100) are read in

LZW Algorithm
Summary



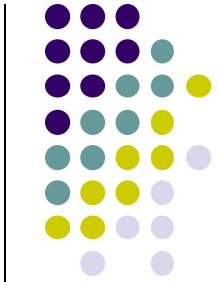
- LZW algorithm is simple – it makes no effort in selecting optimal new strings to enter into its dictionary
- Advantages
  - Works best for streams containing repetitive data
  - Fast
- Disadvantages
  - Files do not contain many repetitive info may actually grow in size after compression
  - Patented technology
- Practical issue
  - Need to limit the size of LZW dictionary – 12-bit codelength
    Dictionary may have up to 4,096 entries
    0-255 reserve for ASCII codes
    The rest for new codes

LZW Algorithm
Issues



- LZW compression is used in UNIX compress, TIFF & GIF files, and zip
  - GIF limits the dictionary size to 12-bit, or 4,096 entries
  - UNIX Compress works on 9-16 bit size codelength
    Start with 9, if filled-up, increase codelength by 1, repeat until codelength = 16
- When code entry is full, LZW becomes static
  - How to ensure code entries are adaptive and up-to-date??
  - UNIX compress re-initialize when compression ratio falls below a threshold
  - May replace LRU (least recently used) entries by new codes found

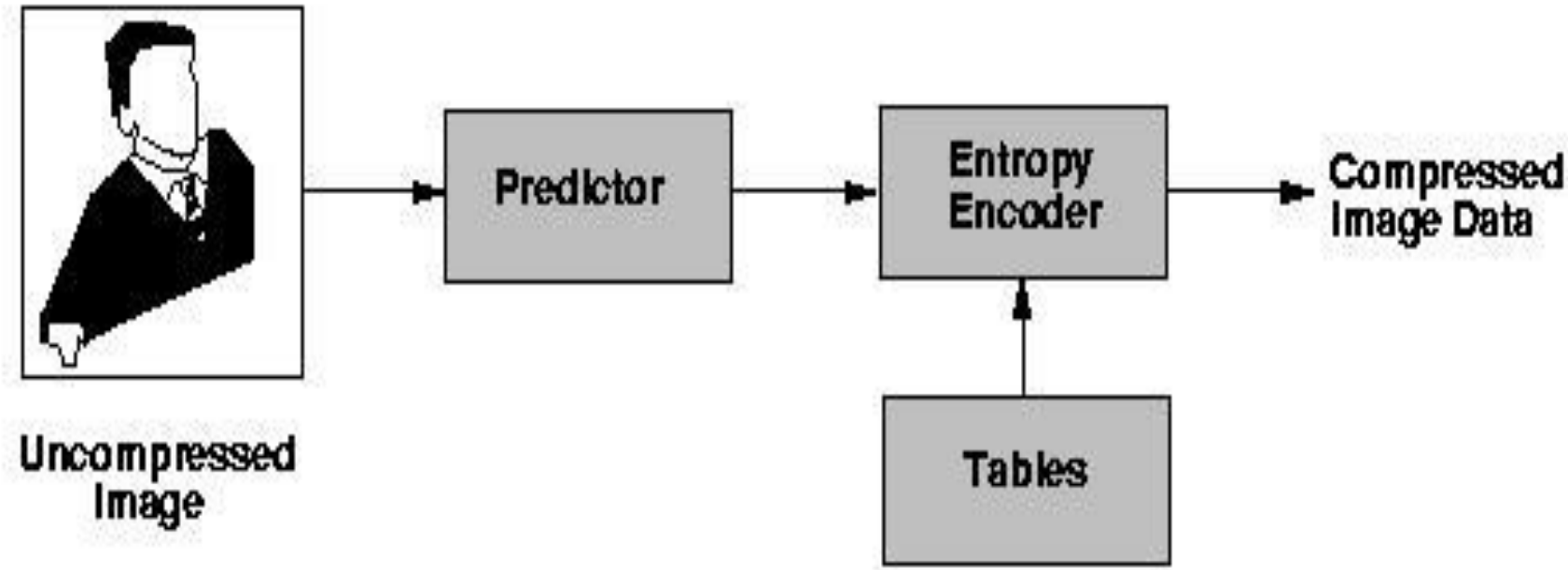
Outline



- Basic Digital Compression
- Huffman & Arithmetic Coding
- LZW Coding
- Predictive Encoding Scheme

Predictive Encoding Scheme

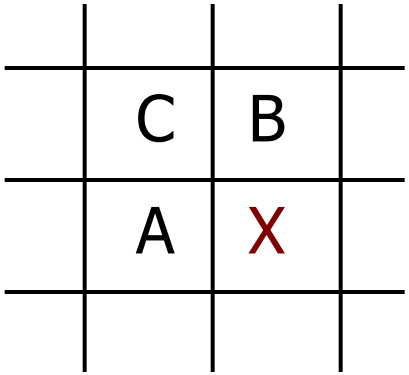
- One simple lossless predictive encoding scheme that operates at the pixel level, rather than (8x8) block level
- Uses a predictive technique to remove statistical redundancy in data



Predictive Encoding Scheme -2

- Each pixel can be encoded with 2 -- 8 bits
- Eight possible predictor values are defined for each pixel based on the values of the adjacent pixel

Predictor Code	Prediction
0	No Prediction
1	X = A
2	X = B
3	X = C
4	X = A + B -C
5	X = A + (B - C) / 2
6	X = B + (A - C) / 2
7	X = (A + B) / 2



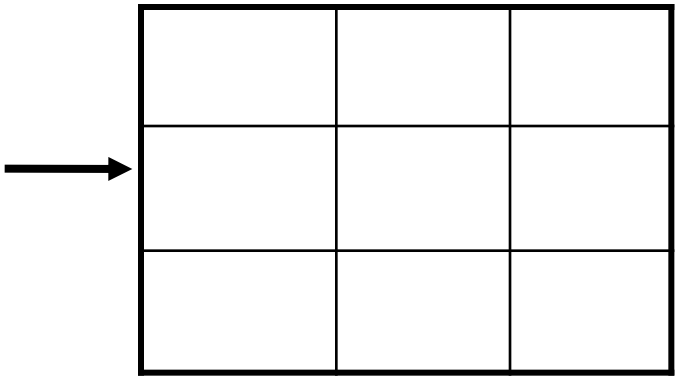
- For each pixel, we then encode:

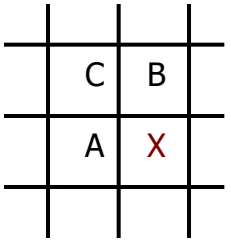
  {Diff = (V_0–X); Code}

Predictive Encoding Scheme -3

- How does it work??? …
- For each pixel: number of the chosen predictor & the difference of the prediction to the actual value are entropy encoded

-76	-73	-67
-65	-69	-62
-66	-69	-60





Predictor Value	Prediction
0	No Prediction
1	X = A
2	X = B
3	X = C
4	X = A + B -C
5	X = A + (B - C) / 2
6	X = B + (A - C) / 2
7	X = (A + B) / 2

- Note, for each pixel, we encode:
  {Diff = (V_0–X); Code}

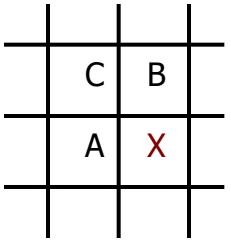
Predictive Encoding Scheme -3

- How does it work??? …
- For each pixel: number of the chosen predictor & the difference of the prediction to the actual value are entropy encoded

-76	-73	-67
-65	-69	-62
-66	-69	-60

→

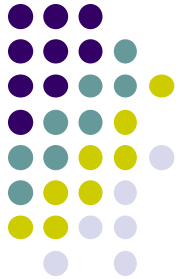
(0, -76)	(1, 3)	(1, 6)
(2, 11)	(7, 0)	(4, 1)
(2, -1)	(…)	(…)



Predictor Value	Prediction
0	No Prediction
1	X = A
2	X = B
3	X = C
4	X = A + B -C
5	X = A + (B - C) / 2
6	X = B + (A - C) / 2
7	X = (A + B) / 2

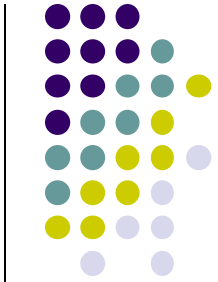
- ?? The predictor and difference values are entropy coded separately
- Given the encoded image, how do you decode

Summary



- Discuss basic compression techniques
  - A range of lossless compression techniques….
- Next Lesson: Lossy compression techniques and JPEG

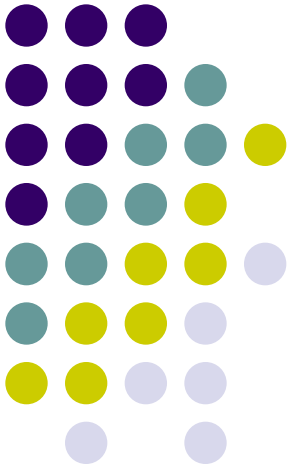
Programming Assignment 2



- Aim:
  - To build a set of Instagram-like filters
- Expected outcome:
  - Negate an image
  - Change brightness of the image
  - Change darkness of the image
  - Style transfer from the image
- Timelines:
  - 26 Oct
- Submission:
  - Source code in Python with following parameters
    - Brightness/ darkness ratio
    - Path for source and target images
    - README

Introduction to Media Computing

Lecture 12: The Lossy Compression Techniques, Image Standards: JPEG



Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

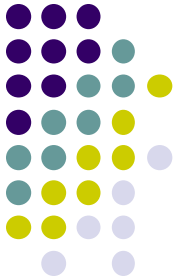
TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

REFERENCES

Main notes extracted from:

- Z.N. Li & M.S. Drew: *Fundamentals of Multimedia*. Prentice Hall, 2004
- G.K. Wallace: The JPEG Still Picture Compression. Communications of the ACM, Vol. 34 (4), pp 30-44, 1991

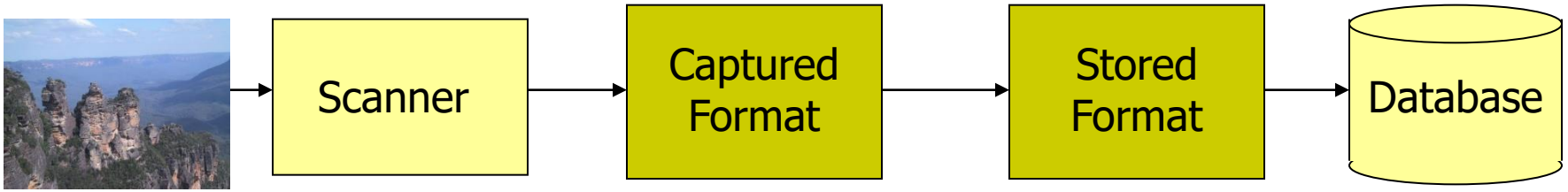
TOPICS



- Image Formats
- Lossy Compression Techniques
  (Chapter 8)
- Monochrome JPEG Standard
  (Chapter 9)

Image Format -1

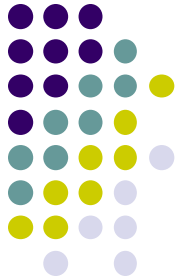
- Captured Image Format (captured through scanner, etc.) -- this is usually the raw image format
- Stored Image Format (storage on disks, CD-ROMs, etc.) -- this is often a compressed image format



Stored Image Format :

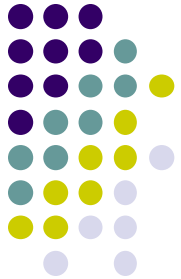
- Store as 2-D array of values
- Meta info stored: width, height, depth (p-values), creator, etc.
- Grayscale: value is 0 – (2^p-1), p = # of bits/pixel
- Color:
  - RGB triplets of pixel color, example $\{r_i, g_i, b_i\}$.
  - Or indexed color, indices to tables of RGB intensities -CLUT

Image Format -2
Common Image Formats



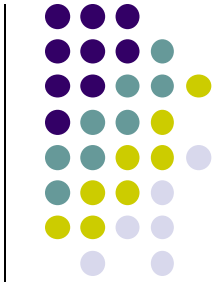
- BMP
  - BMP format supports RGB, Indexed Color, Grayscale, and Bitmap color modes, butdoes not support alpha channels
  - It is a standard Windows image format on DOS and Windows-compatible computers
  - It uses mainly RLE for compression
- TIFF - Tagged-Image File Format
  - Designed by Aldus Corporation and Microsoft in 1987
  - Used to exchange files between applications and computer platforms
  - It uses a lossless compression technique and is suitable for printing applications

Image Format -3
Common Image Formats



- GIF - Graphics Interchange Format
  - It was originally developed by CompuServe in 1987
  - Commonly used to display indexed-color graphics and images in HTML documents over the Web and other online services
  - It uses main LZW for lossless compression

GIF Compression Scheme

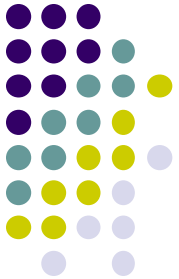


- Used extensively to compress images for use on Internet
- Uses 8-bit format → can represent a max of 256 colors
- Uses a Color Lookup Table (CLUT) to store the color values of pixels.

pixels	R	G	B	index
Pixel1	r1	g1	b1	001
Pixel2	r2	g2	b2	002
Pixel3	r3	g3	b3	003
...	...	...	...	...
Pixel256	r256	g256	b256	256

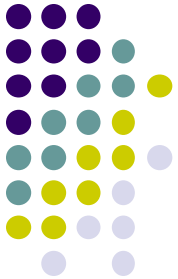
- The compression can both be lossy and lossless
  - Exact if less than 256 colors are used; otherwise it is lossy
  - Further compression is achieved via LZW

Image Format -4
Common Image Formats



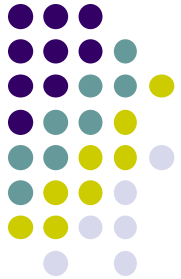
- JPEG - Joint Photographers Experts Group
  - It is commonly used to display photographs and other continuous-tone images in HTML documents over the Web and other online services
  - It Cosine transforms images from spatial to frequency domain to facilitate removal of high frequency components

TOPICS



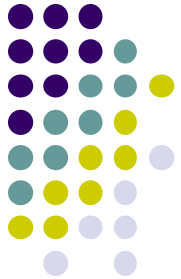
- Image Formats
- Lossy Compression Techniques
- Monochrome JPEG Standard

Introduction



- Lossless compression algorithms do not deliver *compression ratios* that are high enough. Hence, most multimedia compression algorithms are *lossy*.
- What is *lossy compression*?
  - The compressed data is not the same as the original data, but a close approximation of it.
  - Yields a much higher compression ratio than that of lossless compression.

Distortion Measures



- The three most commonly used distortion measures in image compression are:
- *Mean Square Error* (MSE) σ^2,

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

where x_n is input data sequence, y_n the reconstructed data sequence, and N the length of the data sequence

- *Signal to Noise Ratio* (SNR), in decibel units (dB),

$$SNR = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2}$$

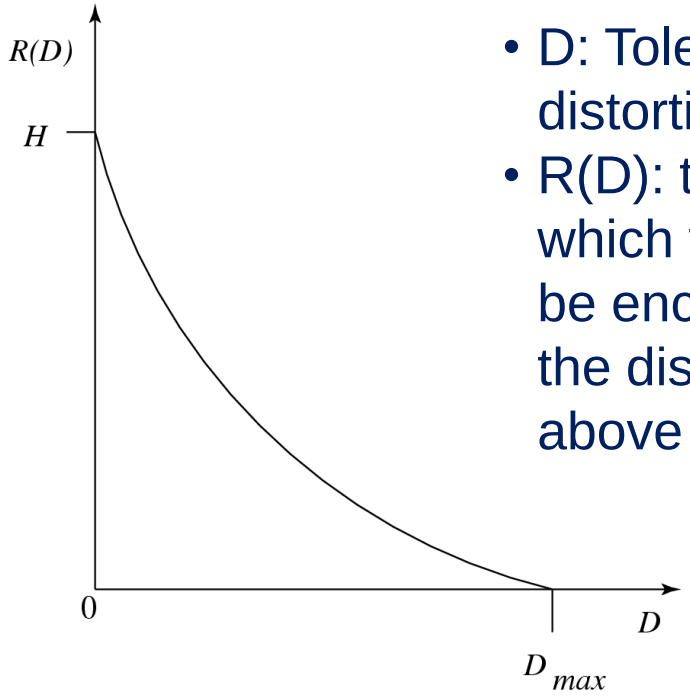
where σ_x^2 is the average square value of the original data sequence and σ_d^2 is the MSE.

- *Peak Signal to Noise Ratio* (PSNR),

$$PSNR = 10\log_{10}\frac{x_{peak}^2}{\sigma_d^2}$$

The Rate-Distortion Theory

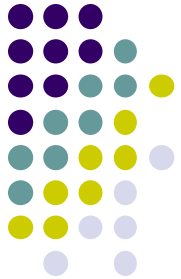
- Provides a framework for the study of tradeoffs between Rate and Distortion.



- D: Tolerable amount of distortion
- R(D): the lowest rate at which the source data can be encoded, while keeping the distortion bounded above by D

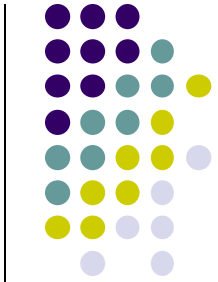
Fig. 8.1: Typical Rate Distortion Function.

Quantization



- Reduce the number of distinct output values to a much smaller set.
- It is the main source of “compression” and hence “loss” in lossy compression
- Three different forms of quantization.
  - Uniform:
  - Nonuniform
  - Vector Quantization

Transform Coding



- Let $\mathbf{X} = [x_1, x_2, .. x_K]^T$ be a vector of input samples
- The rationale behind transform coding:

  Y = T (X), **X, Y**: the input and output vector

  T: a linear transform function

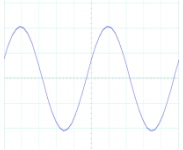
  - If the components of **Y** are much less correlated than that of **X**, then **Y** can be coded more efficiently than **X**
  - If most info is accurately described by the first few components of a transformed vector, then the remaining components can be coarsely quantized, or discarded, with little signal distortion.

- Discrete Cosine Transform (DCT) is able to perform de-correlation of input signal in a data independent manner

Spatial Frequency and DCT

- DCT transforms signals from spatial domain to frequency domain
  - *Spatial frequency* indicates how many times pixel values change across an image block

Low frequency



High frequency

- The DCT formalizes this with a measure of how much the image contents change in correspondence to the number of cycles of a cosine wave per block.
- Hence the role of the DCT is to *decompose* the original signal into its DC and AC components; the role of the IDCT is to *reconstruct* (re-compose) the signal.

Consider a Simple 1D Discrete Cosine Transform First

- Consider the 1D case for simplicity
- 1D Discrete Cosine Transform (1D DCT):

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} f(i)$$

where i = 0, 1, . . . , 7, u = 0, 1, . . . , 7.

- 1D Inverse Discrete Cosine Transform (1D IDCT):
  - The inverse function is almost the same, with the roles of f(i) & F(u) reversed, except that now C(u) must stand inside the sums:

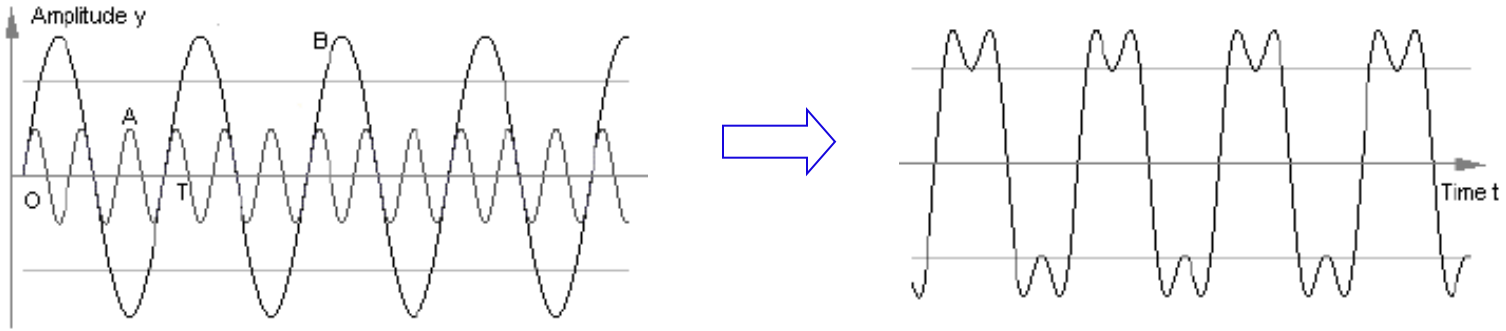
$$\tilde{f}(i) = \sum_{u=0}^{7} \frac{C(u)}{2} \cos \frac{(2i+1)u\pi}{16} F(u)$$

where i = 0, 1, . . . , 7, u = 0, 1, . . . , 7.

- Note: It is transformed into 8 orthogonal frequency bands

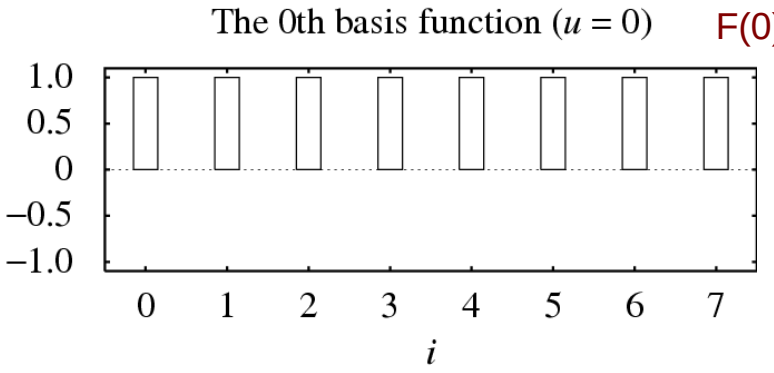
Decomposition of Input Signals to Basic Waveform

- Fourier Transform theory states that any complex waveform can be decomposed into 2 or more basic sinusoidal waves
  - The composite shape is obtained by adding up the amplitudes of the component waves at all points

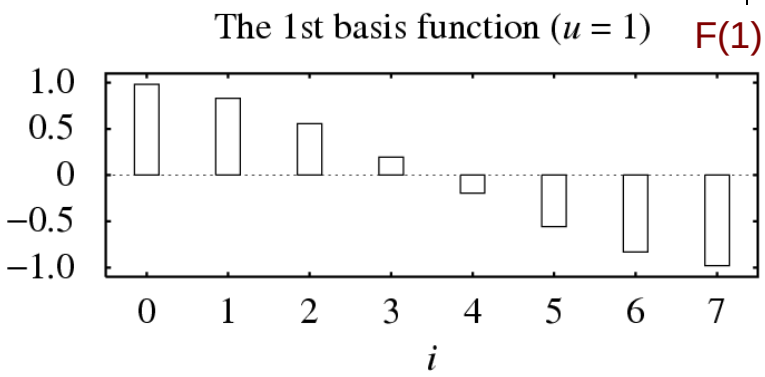


- We want to identify 8 basic wave forms in DCT and determine how an input wave can be decomposed into the basic ones

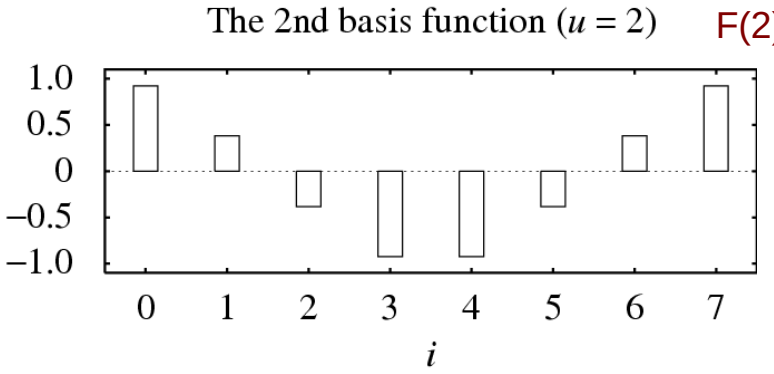
1D DCT Basis Functions -1



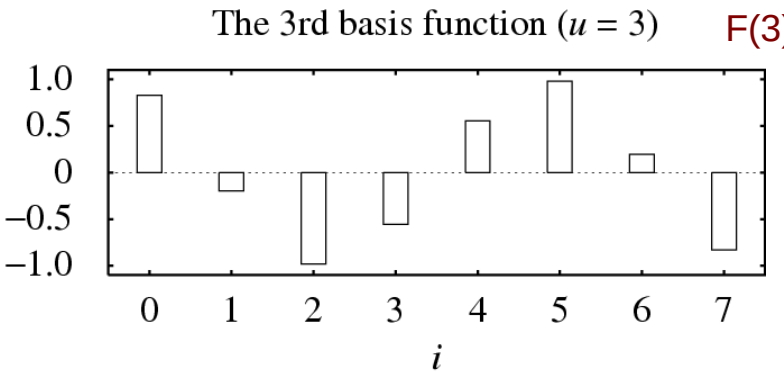
F(0)



F(1)

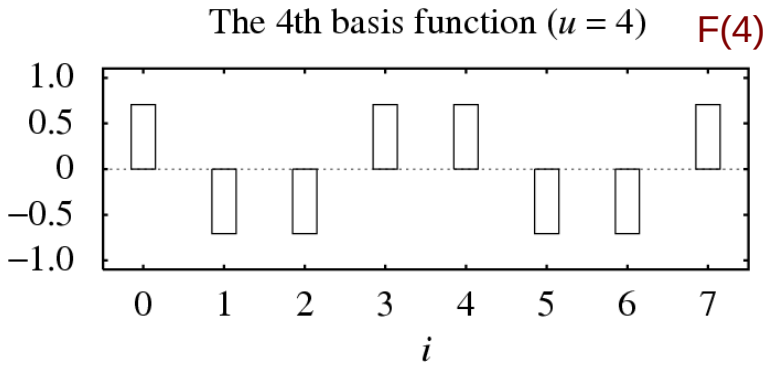


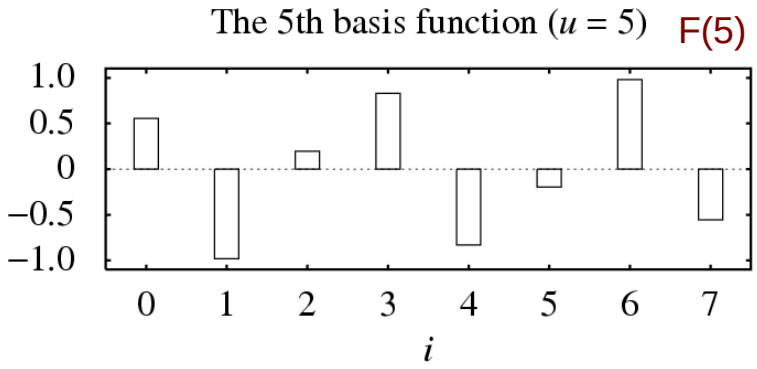
F(2)

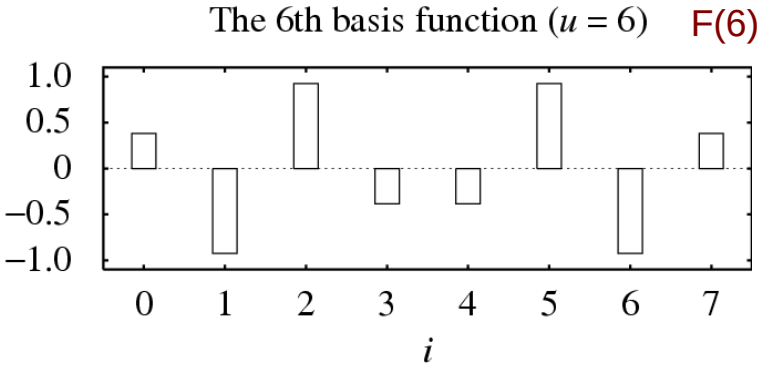


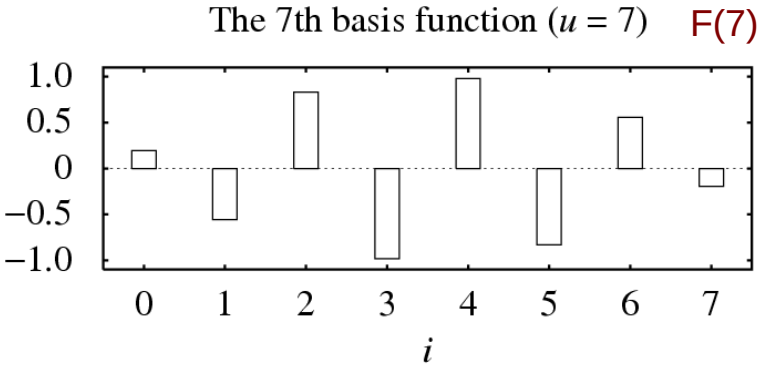
F(3)

1D DCT Basis Functions -2



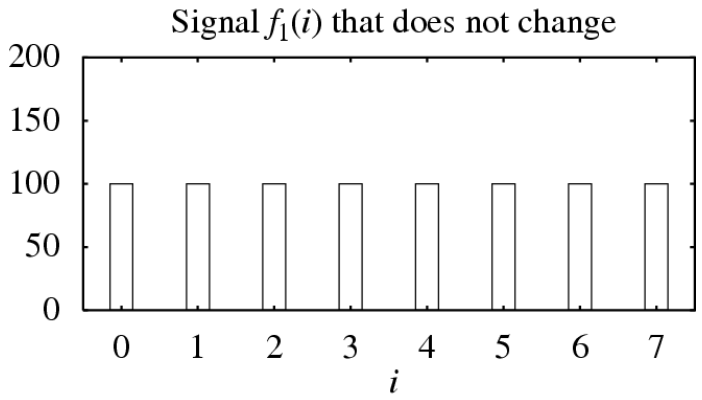




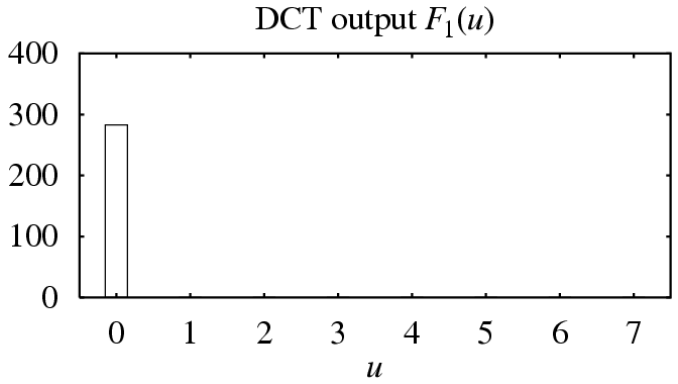


DCT Transform - Example 1

A constant input Signal, f(t) = 100



DCT output, F(u)

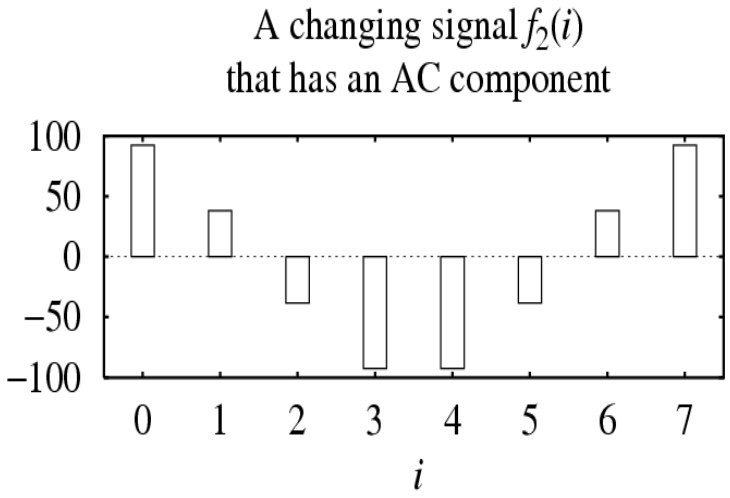


- Maps to F(0) only

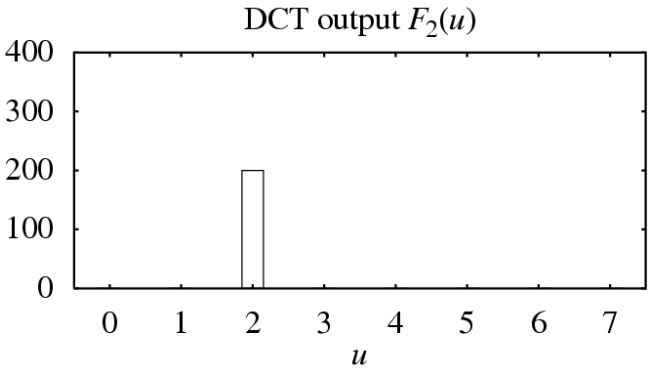
- When u=0, *C(0)* = √2 / 2, and
  F(0) = 283
- When u≠0, *C(u)* = 1, and
  F(1), F(2), .. F(7) = 0
- Note that only *F(0)* is non zero, which is equal to average value of input signal times *C(0)*

DCT Transform - Example 2

A Consine input Signal



DCT output, F(u)



- Maps to F(2) only

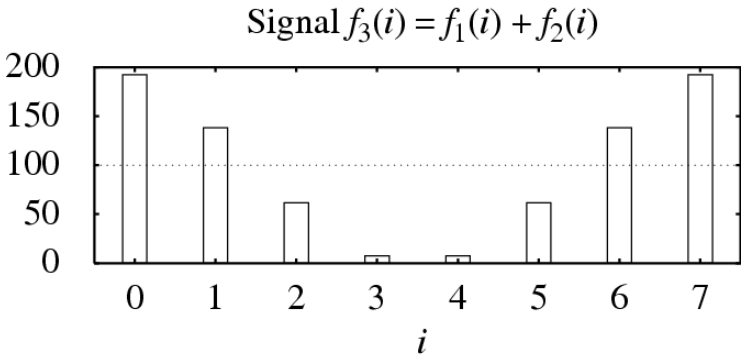
- When u=0, $C(0) = \sqrt{2} / 2$, and $F(0) = 0$
- When u≠0, $C(u) = 1$, and

  $F(2) = 200$

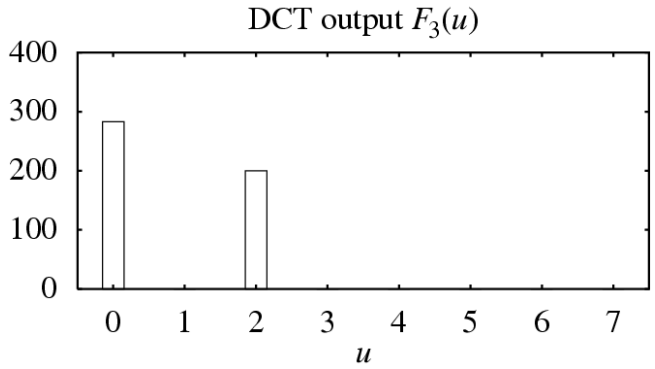
  $F(1), F(3), .. F(7) = 0$
- Note that only $F(2)$ is non zero, which correspond to 2nd basis function of DCT Transform (in fact $f_2(i)$ has the same frequency and phase as 2nd basis function)

DCT Transform - Example 3

Set Input = $f_1(t) + f_2(t)$



DCT output, F(u)



- Maps to F(0) & F(2) only

- We found:

  F(0) = 283

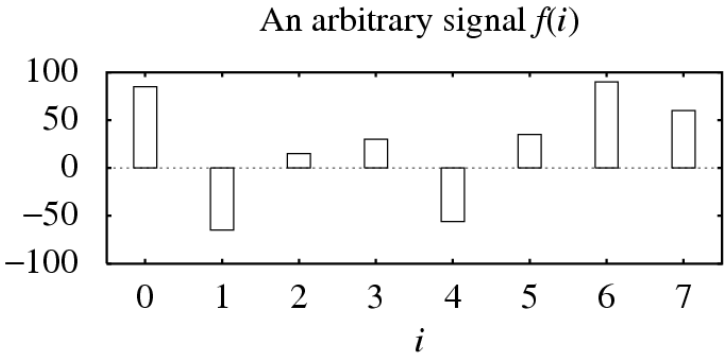
  F(2) = 200

  F(1), F(3), .. F(7) = 0

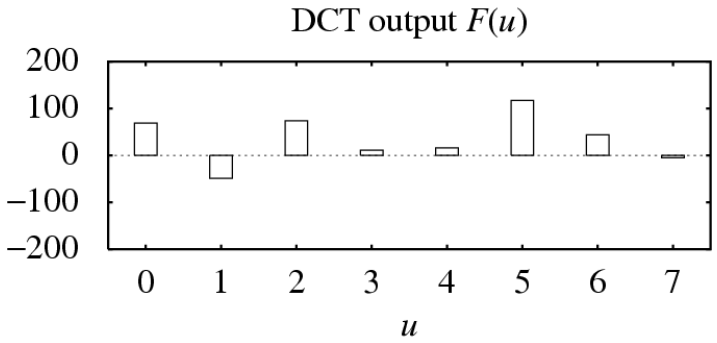
- We discover that $F_3(u) = F_1(u) + F_2(u)$

DCT Transform - Example 4

A complex input Signal



DCT output, F(u)



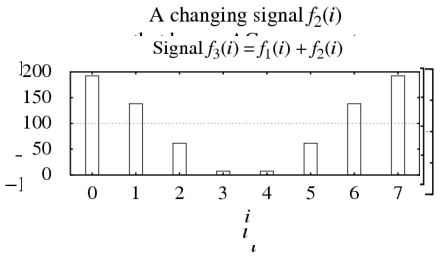
- Maps to a range of F(x) values..

- *f(i)* = [85, -65, 15, 30, -56, 35, 90, 60]
- *F(u)* = [69, -49, 74, 11, 16, 117, 44, -5]

- Note that the DC and a range of AC's are non-zero, reflecting the various frequencies appear in input signals

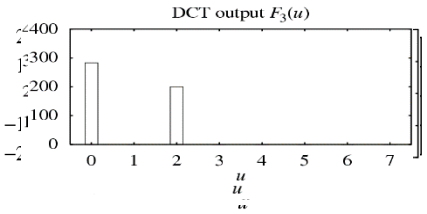
Overall- 1D DCT Transforms

Input Signal, *f(i)*
(in physical domain)



DCT
Transformation

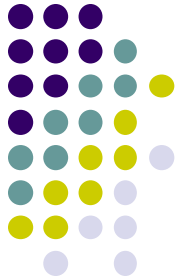
Output Signal, *F(u)*
(in frequency domain)



The outputs are:

- $F_1(u)$ = [283, 0, 0 , 0, 0, 0, 0, 0]
  $F_2(u)$ = [0, 0, 200, 0, 0, 0, 0, 0]
  $F_3(u)$ = [283, 0, 200, 0, 0, 0, 0, 0]
  $F_4(u)$ = [69, -49, 74, 11, 16, 117, 44, -5]
- Note the output *F(u)* depends on spatial frequency of input signals

Observations:

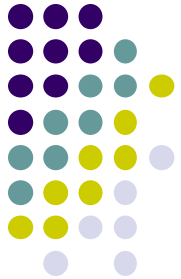


- DCT produces the frequency spectrum *F(u)* corresponding to the spatial signal *f(i)*
  - *F(0)* is DC value; while *F(1)..F(7)* correspond to AC values
- DCT is a linear transform. It satisfies the property:

$$T(\alpha p + \beta q) = \alpha T(p) + \beta T(q)$$

  where α and β are constants, p and q are any functions, variables or constants
  - this property can readily be proven for the DCT because it uses only simple arithmetic operations
- DCT is an Orthogonal transform

Extension to 2D DCT:



- Given an input function $f(i, j)$, the 2D DCT transforms it into a new function $F(u, v)$, with integer u and v running over the same range as i and j.
- The general definition of the transform is:

$$F(u,v) = \frac{2C(u)C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)\cdot u\pi}{2M} \cdot \cos\frac{(2j+1)\cdot v\pi}{2N} \cdot f(i, j)$$

where $i, u = 0, 1, \ldots, M-1$; $j, v = 0, 1, \ldots, N-1$; and the constants $C(u)$ and $C(v)$ are determined by:

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & \text{if} \quad \xi = 0, \\ 1 & \text{otherwise.} \end{cases}$$

2D Discrete Cosine Transform

- 2D Discrete Cosine Transform (for MxN = 8x8):

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

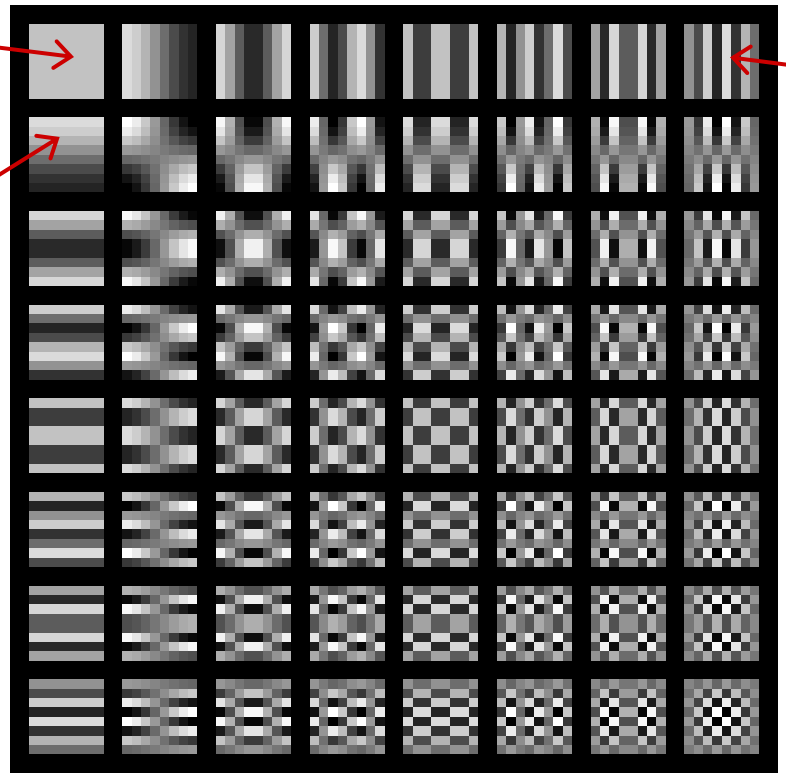
- 2D Inverse Discrete Cosine Transform (2D IDCT):
  - The inverse function is almost the same, with the roles of $f(i, j)$ and $F(u, v)$ reversed, except that now $C(u)$ & $C(v)$ must stand inside the sums:

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

where $i, j, u, v = 0, 1, \ldots, 7$.

Interpretation of 2D DCT Transform

- What does the the 8x8 DCT coefficients look like?

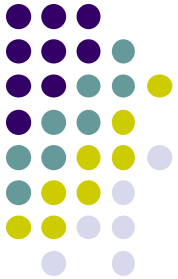


- Top line accounts for vertical variations
- The DCT features of an image
- DC coeff at top-left
- 63 AC images of varying frequencies

- DC accounts for contents (down-sampled)
  - Select only DC values → an image scaled down by 8 in vert. & hor. directions
- AC values account for contrast in image → represent contrast or edge-ness in each block by magnitude of selected AC values

Overall- 2D DCT Transforms

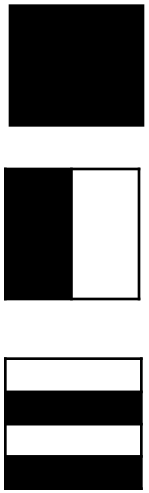
Demo: http://cgjennings.ca/toybox/hjpeg/index.html

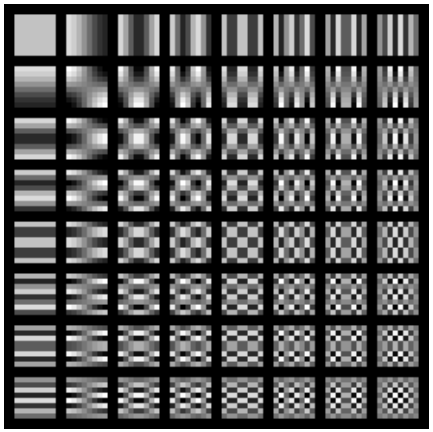


Input Signal, *f(I,i)*
(in physical domain)

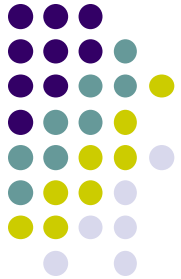
DCT
Transformation

Output Signal, *F(u,v)*
(in frequency domain)





DCT Transform:
2D Separable Basis



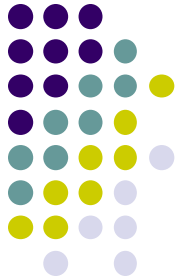
- The 2D DCT can be *separated* into a sequence of two, 1D DCT steps:

$$G(i,v) = \frac{1}{2} C(v) \sum_{j=0}^{7} \cos \frac{(2j+1)v\pi}{16} f(i,j)$$

$$F(u,v) = \frac{1}{2} C(u) \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} G(i,v)$$

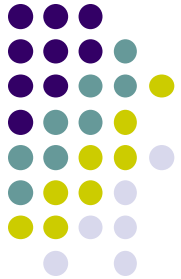
It is straightforward to see that this simple change saves many arithmetic steps. The number of iterations required is reduced from 8 × 8 to 8 + 8.

TOPICS



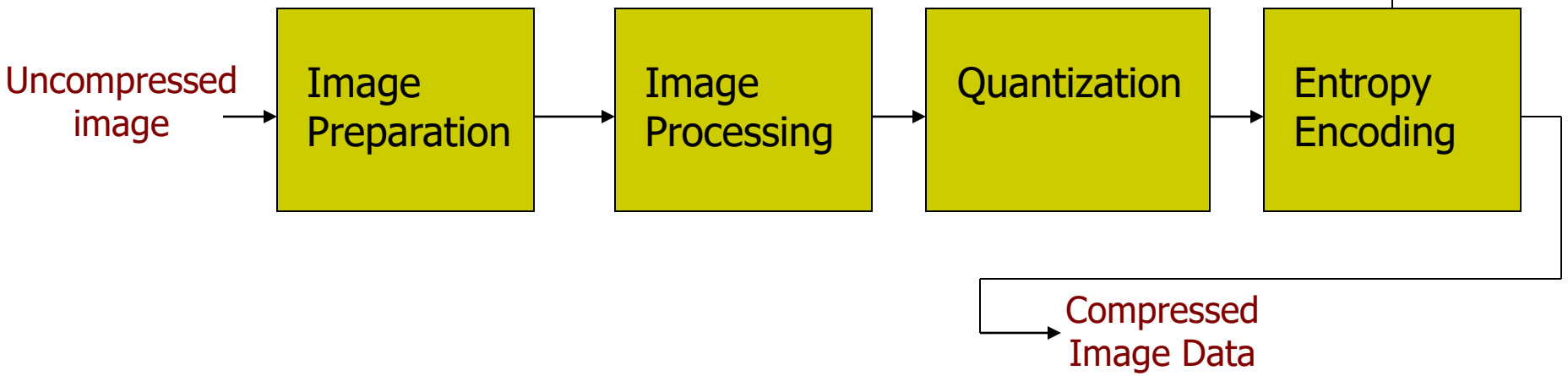
- Image Formats
- Lossy Compression Techniques
- Monochrome JPEG Standard

JPEG Compression for Monochrome Images



- JPEG: Joint Photographers Experts Group
- Established in 1986 and standard was adopted in 1991
- JPEG applies to gray-scale and color images
- JPEG Committee had these requirements:
  - Independent of image size
  - Support any content complexity
  - Good compression ratio without sacrificing image quality
  - Run on most standard platforms
  - Have both sequential decoding and progressive decoding abilities
- MJPEG (Motion JPEG) refers to a video format in which each frame is coded in a JPEG mode

Compression Phases



- **Image Preparation:** Analog-to-Digital conversion
- **Image Processing:** Transform content from pixel to frequency domain
- **Quantization:** Map pixel values to integer values with limited range
- **Entropy Encoding:** Last step -- compress sequential data without loss. (e.g., run-length coding or Huffman coding)

Block Diagram of JPEG Encoder

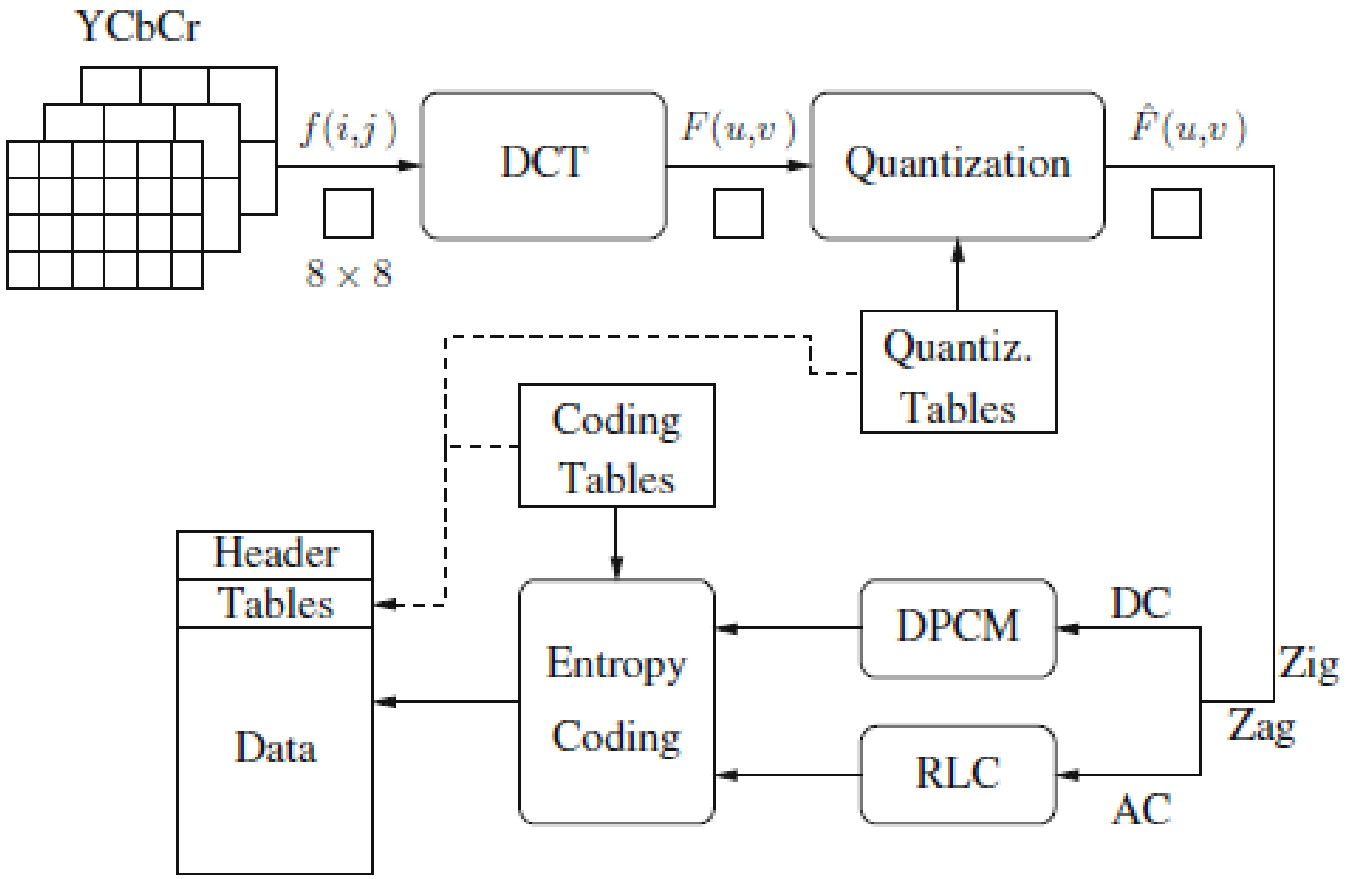
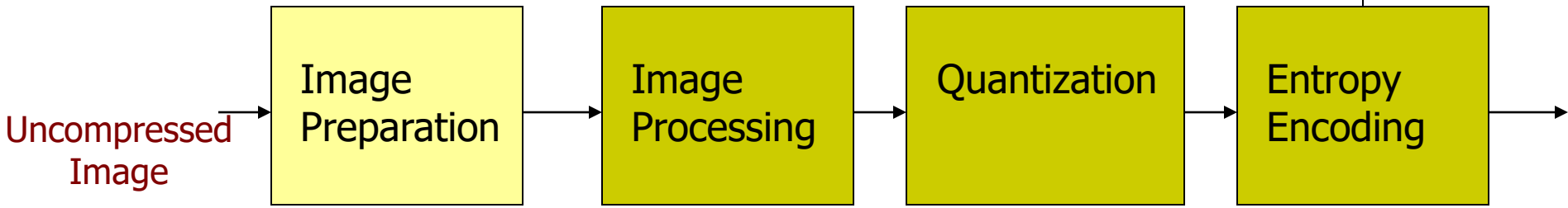


Image Preparation -1



- Consider grayscale image first (with p bits)
- It is sufficient to say at this stage that we divide the image into 8x8 blocks, shifted from

  unsigned integers with range $[0, 2^p]$

  → signed integers with range $[-2^{p-1}, 2^{p-1} -1]$

  - Effect: fix the nominal dynamic range of DC values to be centered about zero
- Next: perform transform coding on each 8x8 block

Image Preparation -2

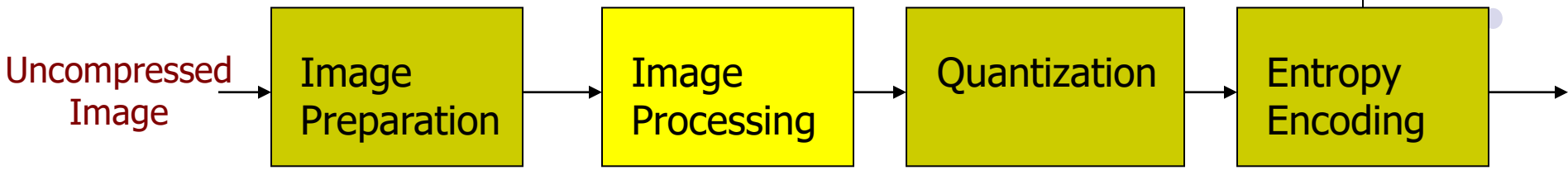
- Given an original 8x8 sub-image. Its shifted array is (by $2^{p-1} = 128$):

52	55	61	66	70	61	64	73
63	59	66	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

→

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

DCT Transform -1



- Perform DCT on each 8x8 blocks of pixels
- The 8x8 forward DCT is defined as:

$$F(u,v) = C(u)C(v)[\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y) * \cos\frac{(2x+1)u\pi}{2N} * \cos\frac{(2y+1)v\pi}{2N}]$$

where C(u), C(v) = √ (1/N) for u, v = 0

C(u), C(v) = √(2/N) otherwise, for DCT: N=8

- FDCT maps the values from spatial domain (8x8 pixels) to frequency domain (64 orthogonal basis functions)
  - Consists of 1 DC value and 63 AC values

DCT Transform -2

JPEG is a lossy image compression method. The effectiveness of the DCT transform coding method in JPEG relies on three major observations:

Observation 1. Useful image contents change relatively slowly across the image—that is, it is unusual for intensity values to vary widely several times in a small area—for example, in an 8×8 image block. Spatial frequency indicates how many times pixel values change across an image block. The DCT formalizes this notion with a measure of how much the image contents change in relation to the number of cycles of a cosine wave per block.

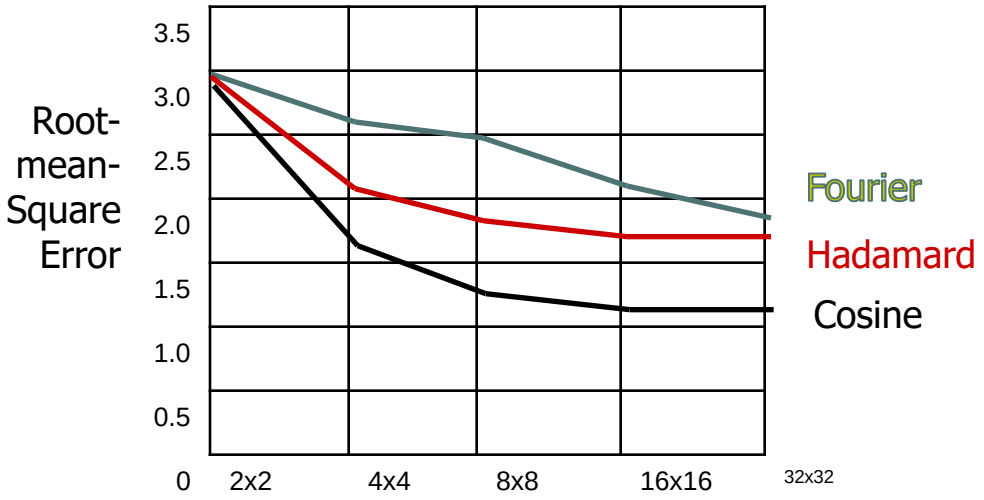
Observation 2. Psychophysical experiments suggest that humans are much less likely to notice the loss of very high-spatial frequency components than lower frequency components.

Observation 3. Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray ("black and white") than for color. We simply cannot see much change in color if it occurs in close proximity—think of the blobby ink used in comic books. This works simply because our eye sees the black lines best, and our brain just pushes the color into place. In fact, ordinary broadcast TV makes use of this phenomenon to transmit much less color information than gray information.

DCT Transform -2

- WHY 8x8 Blocks
- Choice of N
  - Reduce coding error (larger N --> smaller error)
  - Computational complexity (larger N --> more complex)
- Compromise of all requirements
- N=8 gives best compromise between error & complexity

Tested on three transform methods:



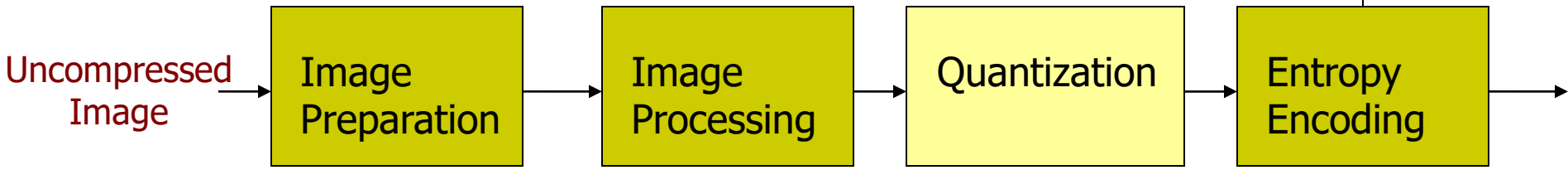
DCT Transform -3

- Continue with the shifted sub-image, its FDCT values are:

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

Quantization -1



- How does quantization work?
  - Given $x_1 = 54$, $x_2 = 36$
  - Quantization steps: $q_1=10$ ➔ equivalent round to nearest multiple of 10

    Integer Round (x/q): $x_1 \rightarrow I(54/10) = 5$, $x_2 = 4$

    Reverse process: $x' = x*q$: $x'_1 \rightarrow 5*10 = 50$ (error= 4)

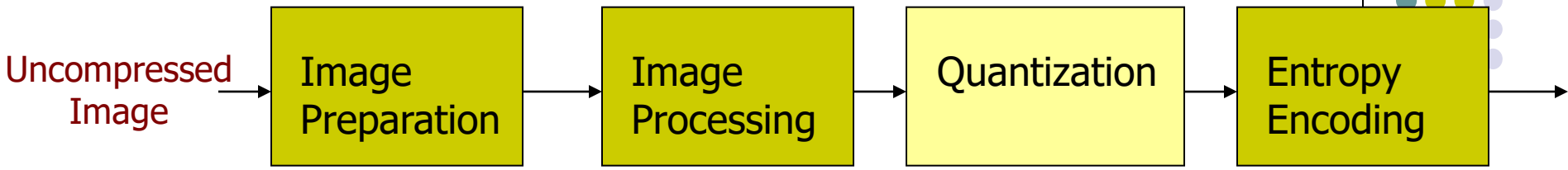
    $x'_2 \rightarrow 4*10 = 40$ (error= 4)
  - Quantization steps: $q_2=5$ ➔ equivalent round to nearest multiple of 5

    Integer Round (x/q): $x_1 \rightarrow I(54/5) = 11$, $x_2 = 7$

    Reverse process: $x' = x*q$: $x'_1 \rightarrow 11*5 = 55$ (error= 1)

    $x'_2 \rightarrow 7*5 = 35$ (error= 1)
  - Smaller quantization step ➔ greater accuracy, less error, but less compression

Quantization -2



- Uses psycho-visual analysis to decide how much to quantize and where in the 8x8 DCT array
  - In general, smaller quantization steps for lower frequency components
- One example of quantization Table stored in JPEG file

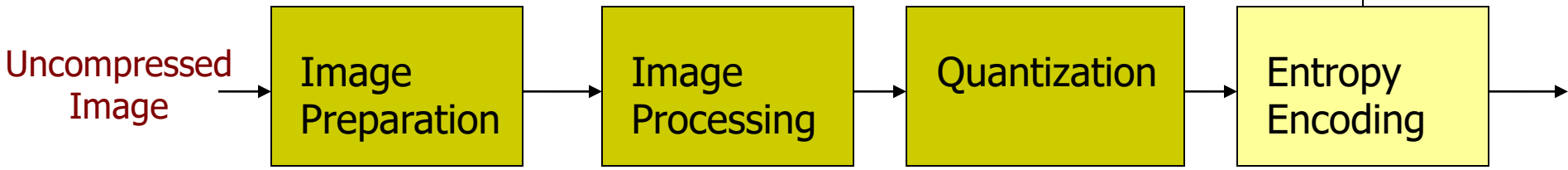
The *Luminance Quantization Table* q(u, v)

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

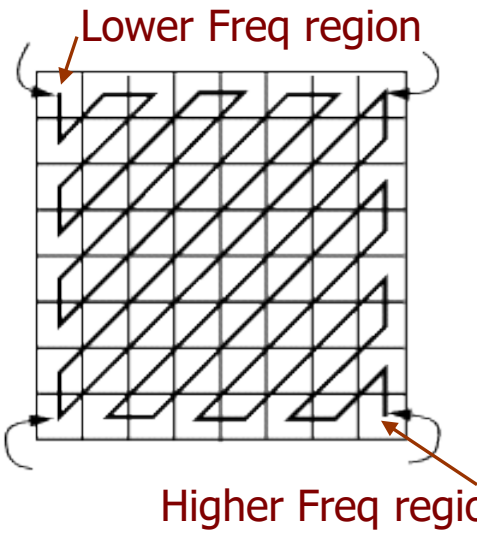
→

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	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

Entropy Coding -1



- After appropriate quantization, most DCT coeffs (esp. those corresponding to higher frequency) will be zero
- Order the coeffs in zig-zag sequence shown (most trailing coeffs will be zero)



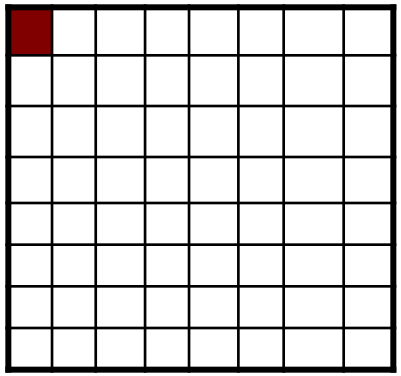
-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

➔ [-26, -3, 1, -3, -2, -6, 2, -4, 1, -4, 1, 1, 5, 0, 2, 0, 0, -1, 2, 0, 0, 0, 0, 0, -1, -1, EOB]

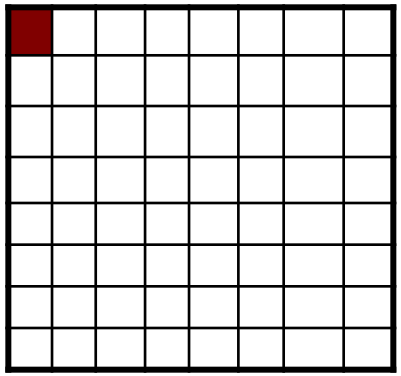
Entropy Coding -2

- We can produce further compression by reducing the size of data to be coded
- As DC values are quite large, they are coded differently from ACs
  - DC-coefficients are encoded as difference of the current DC-coefficient DC_i and the previous one, or
  - Encoding of DC value for frame i is: $DC_i - Dc_{i-1}$
  - Note: DC DCT coefficients of adjacent blocks are highly correlated.

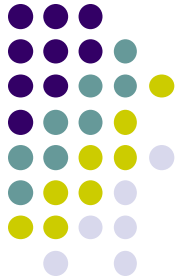
DC_{i-1} of Frame i-1



DC_i of Frame i

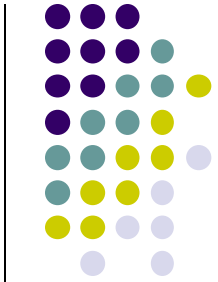


Entropy Coding -3



- Next step : use run-length encoding of zero values.
  follow by Huffman or Arithmetic Coding
- JPEG specifies that either Huffman or Arithmetic coding (protected by patents) can be used. Huffman encoding is most commonly

Image Decoding -1
Simply reverse the process



- Decode Huffman/Arithmetic coded data, follow by reverse RLE
- Dequantize the data : $F^R(u,v) = F^Q(u,v) \times Q(u,v)$

  The same quantization matrix Q which was used in the encoding process should be used here
- Dequantized coefficients are then inverse transformed – by applying the inverse discrete cosine transform (IDCT)

Image Decoding -2
Inverse DCT Transform

- There is a corresponding 8x8 inverse DCT is defined as:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) * \cos\frac{(2x+1)u\pi}{2N} * \cos\frac{(2y+1)v\pi}{2N}$$

which maps the compressed image in DCT representation to spatial (pixel) representation for display

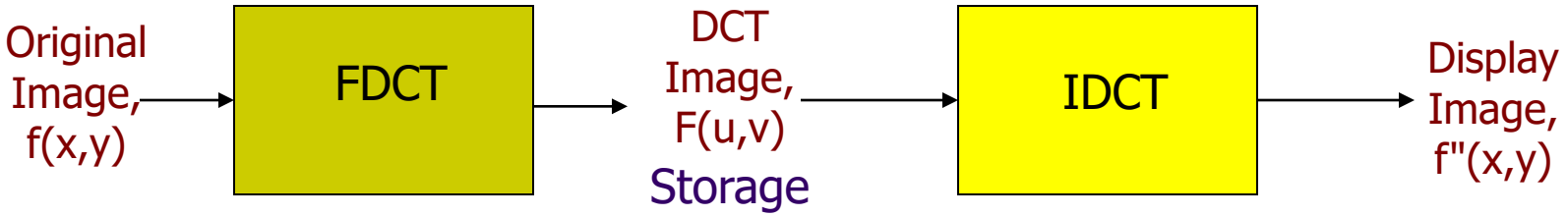
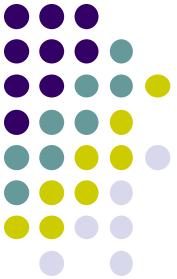
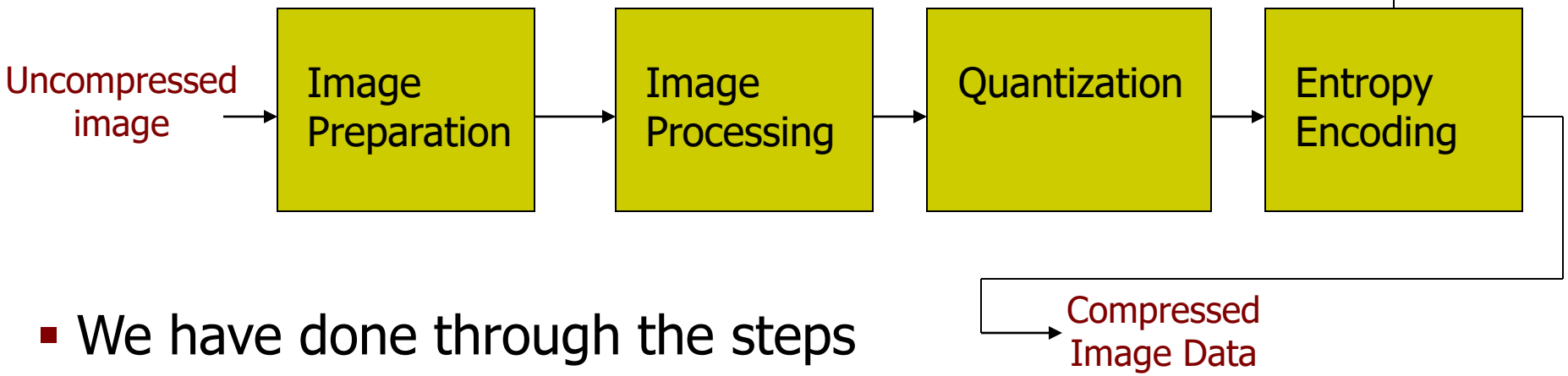


Image Decoding -3
Overall



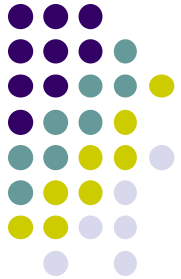
- DCT Transform is lossless
- While quantization step is lossy
- Note FDCT and IDCT involve some loss of information due to limited precision
- JPEG specifies compliance testing by using an official set of test images and corresponding distortion measure
- Main Question: Where does the major compression take place?

Summary



- We have done through the steps for an image with single components (could be color too based on CLUT)
- The stages for images with more components, for example a color image with 3 components, will be more complex
- We will examine these steps for color images next

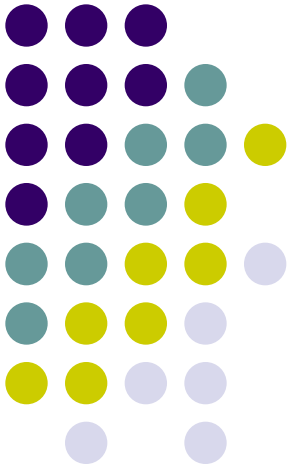
Summary



- Discuss basis of compression in JPEG
- Next Lesson: Color Models, Color JPEG, and other Advanced Image Compression proposals

Introduction to Media Computing

Lecture 13: Color Model and Color JPEG

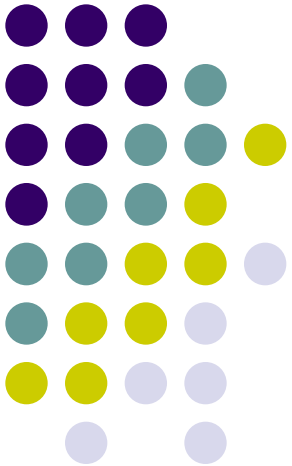


Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

Introduction to Media Computing

Lecture 13: Color Model and Color JPEG



Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

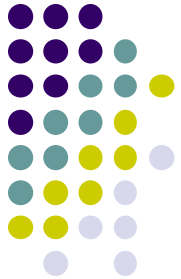
TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

REFERENCES

Main notes extracted from:

- Z.N. Li & M.S. Drew: *Fundamentals of Multimedia*. Prentice Hall, 2004
- G.K. Wallace: The JPEG Still Picture Compression. Communications of the ACM, Vol. 34 (4), pp 30-44, 1991

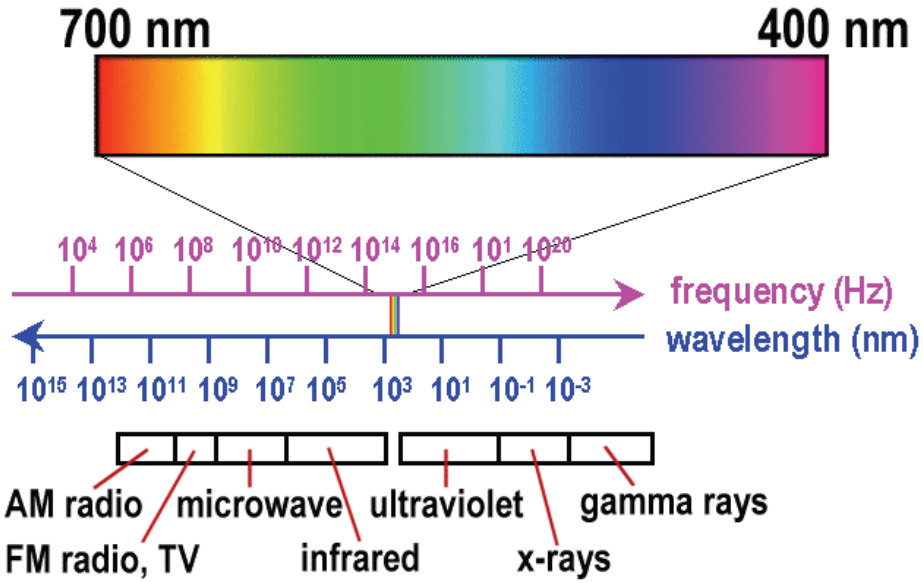
TOPICS



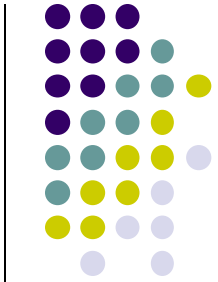
- The Color Models (Chapter 4)
- Color JPEG Standard (S 9.1.1)
- JPEG Modes (S 9.1.2)
- New Image Compression Proposals

Color

- Color is a powerful descriptor that helps simplifies object identification and extraction from a scene
- Humans can discern thousands of color shades and intensities, compared to only about two dozen shades of grays
- Human eyes respond to “visible light”
  - tiny piece of spectrum between infra-red and ultraviolet
- Color defined by the emission spectrum of the light source
  - amplitude vs wavelength (or frequency) plot

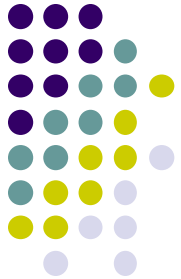


Color Characteristics -1



- We can perceive colors in terms of brightness, hue & saturation
- Brightness
  - Color intensity
- Hue
  - Represents dominant color as perceived by an observer
  - We call an object red, orange, or yellow …
- Saturation
  - Refers to relative purity or amount of white light mixed with a hue
  - The pure spectrum colors are fully saturated
  - Colors such as pink (red and white) and lavender (violet and white) are less saturated
  - Degree of saturation is inversely proportional to the amount of white light added

Color Characteristics -2



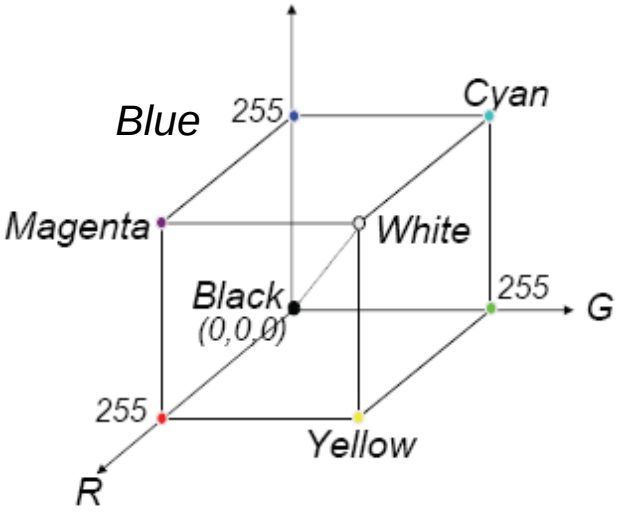
- Hue + Saturation ⇒ Chromaticity
- Brightness: Gray scales
- Color : Chromaticity + Brightness

Hardware Oriented Color Model -1

- A color model is a specification of a coordinate system and a subspace within that system where each color is represented by a single point

RGB Color Model

- Cartesian coordinate representation
- *Full-color* image is used often to denote a 24-bit RGB color image
- Used by most digital imaging devices (monitors & color cameras)

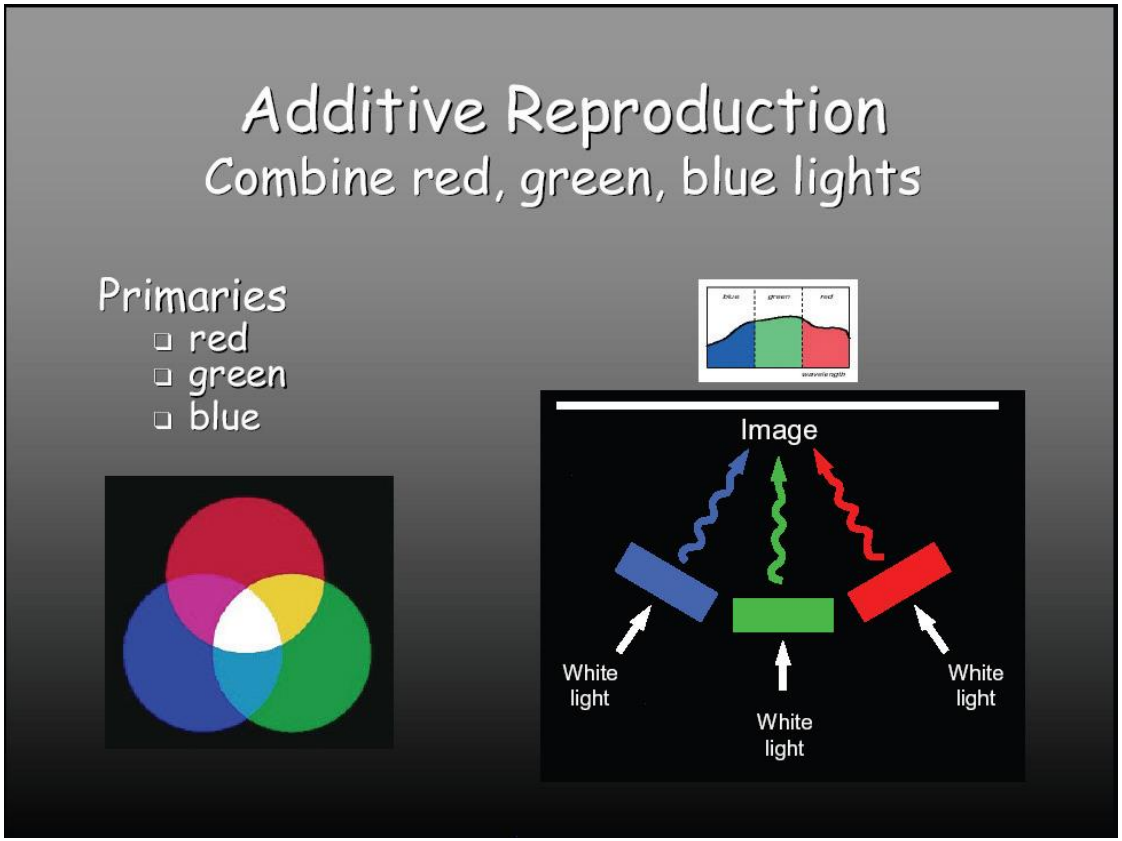


Problems with RGB Colors:

- It is nonlinear, not well suited for describing colors by humans
- What is the meaning of nonlinearity?

Hardware Oriented Color Model -2

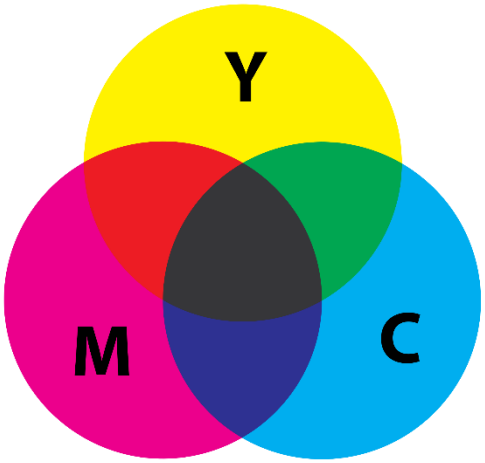
- RGB is termed as “additive Color model”, as the resulting color, such as the “white” is the "additive" combination of all primary colored lights



Hardware Oriented Color Model -3

- CMY(K) Color Model:
  - Designed for color printing
  - **CMYK**: cyan, magenta, yellow and key (black).
  - When CMYK “primaries” are combined at full strength, the resulting “secondary” mixtures are red, green and blue.

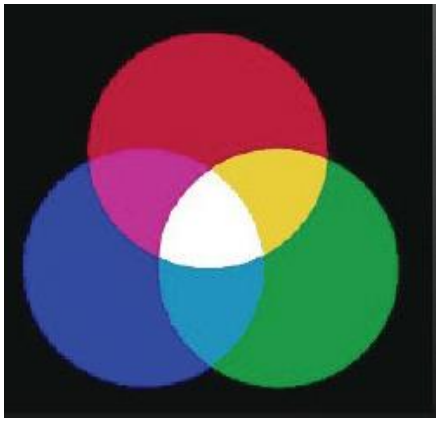
    Mixing all three gives an imperfect black.



CMY

  - In contrast to RGB, it is a subtractive color model
  - Why it is called subtractive?

    Because inks "subtract" brightness from white while black is the absence of light



RGB

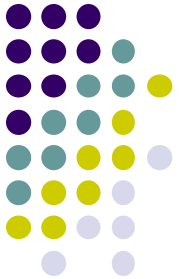
TV Broadcast Models -1

- TV Broadcast Color Models
  - Examples are: YUV, YCrCb
  - YUV were designed for encoding TV signals, while YCrCb for video & still image compression and transmission

- YUV Model:
  - Y: Luminance or Black-and-White (B&W) component
  - U & V: Chrominance or color components
  - Was invented to broadcast color TV in a B&W infrastructure. The model is compatible with B&W TV while being able to add color.
  - Basic color format used by the NTSC, PAL, SECOM
  - U and V sub-sampled to reduce bitrate

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

TV Broadcast Models -2



- YCrCb Model
  - Is used for *digital encoding* of color info suited for video *and* still-image compression and transmission such as MPEG and JPEG
  - Y: Luminance component
  - Cr & Cb: Chrominance or color components
  - Used in digital image/video compression standards

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & 0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

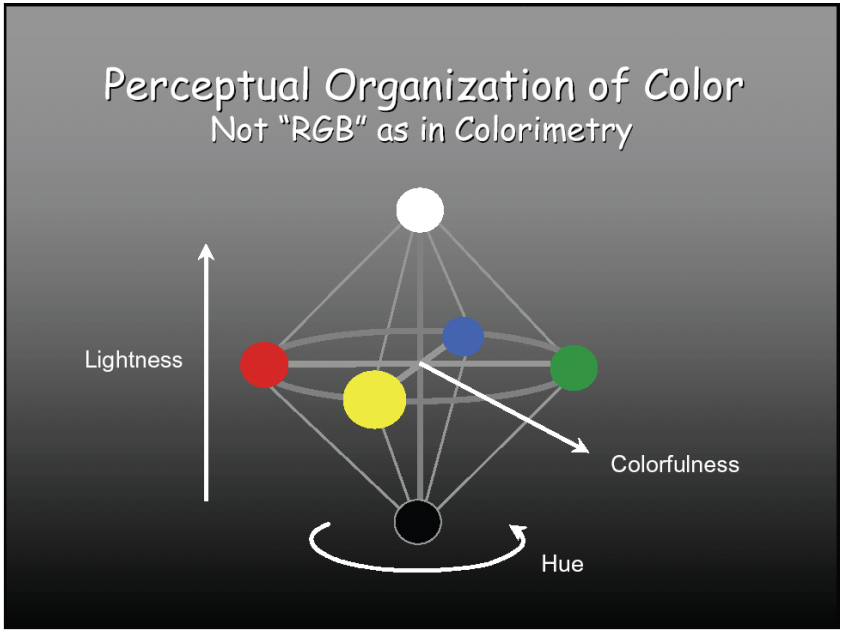
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & -0.71414 \\ 1.0 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

Human-Oriented Models -1

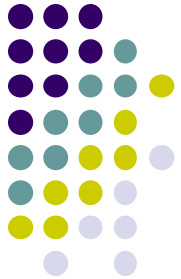
HSI, HSL, HSV Family

- The hardware and broadcast oriented color models are not well suited for describing colors in terms that are practical for human interpretation
- As humans view a color object in terms of its hue, saturation and brightness

- Brightness is a subjective descriptor that is impossible to measure
- Hue, Saturation and Intensity/Lightness/Value
  - These model describes a color in terms of how it is perceived by the human eye
  - more intuitive in manipulating colors

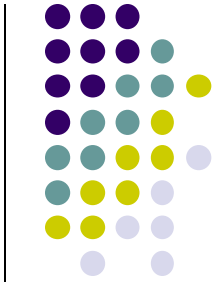


Human-Oriented Models -2



- Perceptually uniform color space
- Other Models
  - Munsell, L*a*b, L*u*v
  - Used for color matching & image analysis
  - Has further advantage that color axes are orthogonal

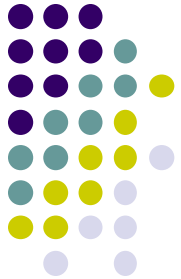
TOPICS



- The Color Models (Chapter 4)
- Color JPEG Standard (S 9.1.1)
- JPEG Modes (S 9.1.2)
- New Image Compression Proposals

Color JPEG -1

Image Preparation



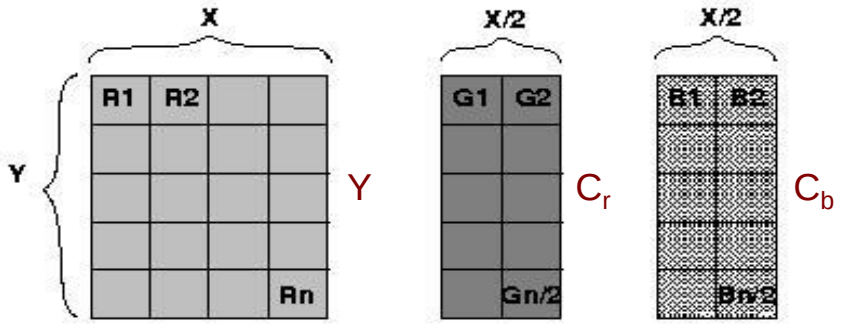
- JPEG adopts YC_rC_b model for color representation
  - Y is the luminance component → grayscale
  - {Cr, Cb} are the color (or chrominance) components
  - **Human is less sensitive to Chrome than to Luminance**
  - Hence we can afford to lose a lot more info in chrominance components than in luminance component

Color JPEG -2

Image Preparation

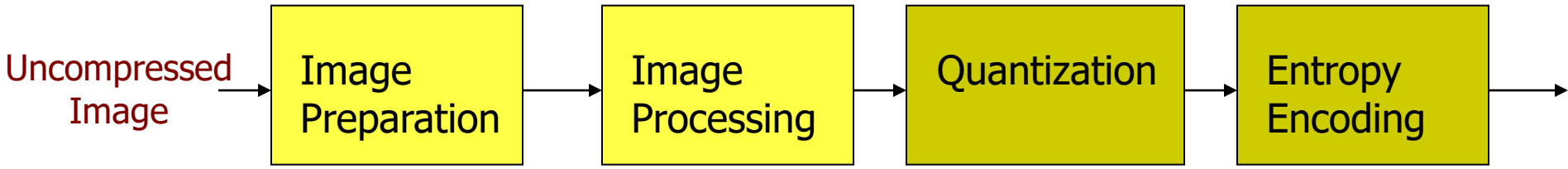
- Typical choice:
  - Leave Luminance component alone at full resolution
  - Chrominance components are often reduced 2:1 horizontally and either 2:1 or 1:1 (no change) vertically
  - We often called these alternatives 2h2v or 2h1v sampling
  - This immediately reduce the data size by 1/2 or 1/3 → 1st phase of compression

- For 2H1V sampling → 1/3 saving
  - $Y_1 = Y_2 = Y_3$; $X_1 = 2X_2 = 2X_3$



Color JPEG -1

Image Preparation & DCT Transform



- These two steps are similar to that in monochrome case, except that the steps are repeated 3 times, 1 for each bit plane

Color JPEG -2

Image Preparation & DCT Transform

- Visualization of Color DCT Transform

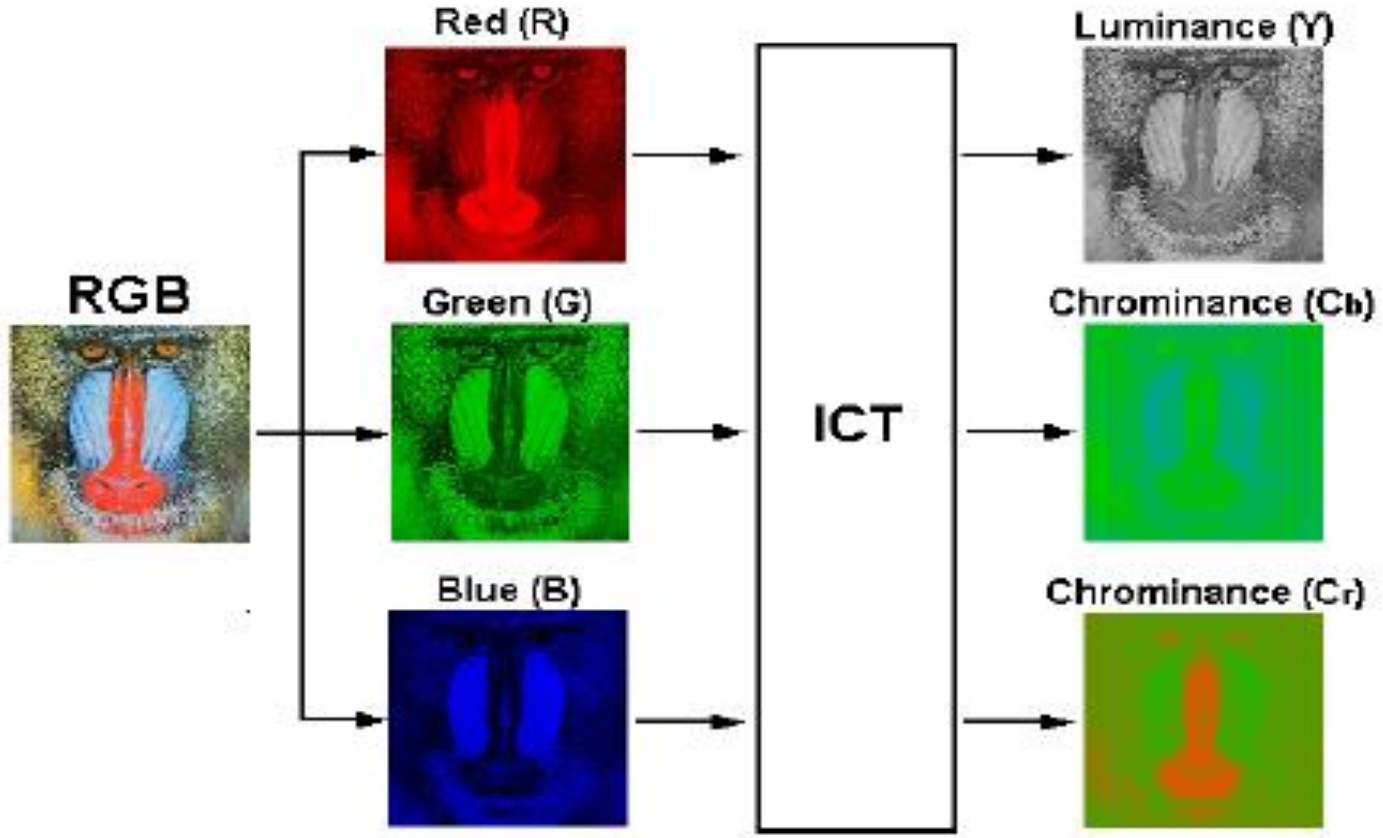
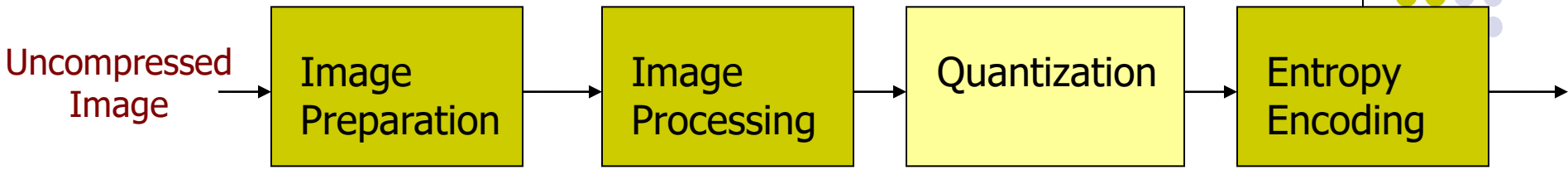


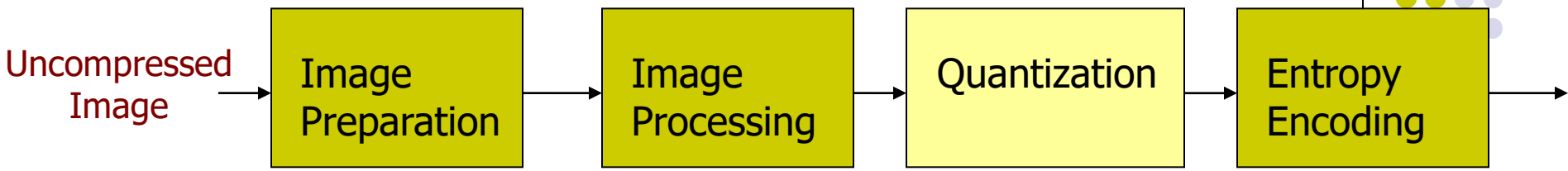
Figure 7: The ICT of the baboon image.

Quantization -1



- How does quantization work? -- uses the Psychovisual Features
- Human Visual System (HVS): Based on studies done by many researchers, we know that visibility of distortion depends on the following image content features:
  - **Luminance vs. Chrominance**: The eye is much less sensitive to the chrominance channels than the luminance channel
  - **Edges**: Edge distortions are noticed easily
  - **Smooth areas**: Distortion of smooth regions is noticed easily
  - **Textures**: The distortion is low for a strong texture and high for a low-texture region
  - **Brightness**: The mid-gray regions have lower distortion sensitivity compared to other gray bands

Quantization -2



- Different types of quantization
  - Constant amount – less effective
  - Varying amount for different frequencies
  - Separate values for luminance and chrominance (larger amount for chrominance)

Quantization -3

- JPEG adopts nonlinear quantization by using the quantization tables
  - Sample quantization table is used – derive from psychophysical studies
  - We can change the compression ratio by using the scale factor, S, to scale the numbers in Q(u,v) matrix
  - F'(u,v) = round [F(u,v) / (S * Q(u,v))]

The *Luminance Quantization Table* q(u, v)

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

The *Chrominance Quantization Table* q(u, v)

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

Note: apply larger quantization for higher frequency AC values

Quantization -4 (Examples)

S = 1
Size = 60.8 KB

S = 2
Size = 12.8 KB

S = 3
Size = 6.9 KB

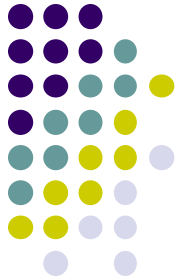
S = 4
Size = 2.6 KB

S = 5
Size = 1.8 KB

S = 6
Size = 1.6 KB

$$F'(u,v) = \text{round}\left[F(u,v) / (S * Q(u,v)) \right]$$

Quantization -5



SUMMARY

- Each coefficient F(u,v) is scaled by a factor S*Q(u,v) (quantization factor).

  In general:

  $$F^{Q}(u,v) = \text{Integer-Round}\,[F(u,v)/(S*Q(u,v))]$$

  - In general, lower frequencies are quantized in a fine manner (i.e., with lower Q values) than the higher frequency ones

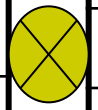
- Dequatization can be done by:

  $$F^{R}(u,v) = F^{Q}(u,v) \times S \times Q(u,v)$$

Quantization -6

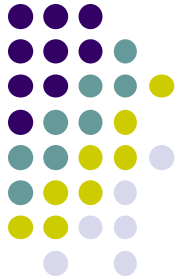
- Continue from previous example, we apply the Q matrix on the output of FDCT step is :

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



-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

Quantization -7

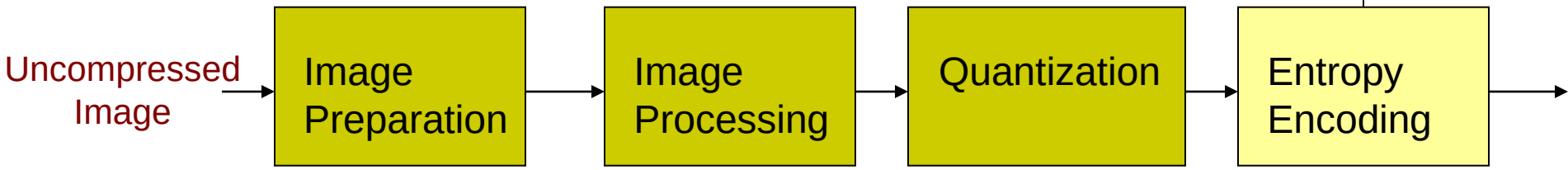


- The results of performing the quantization using the Q matrix is:

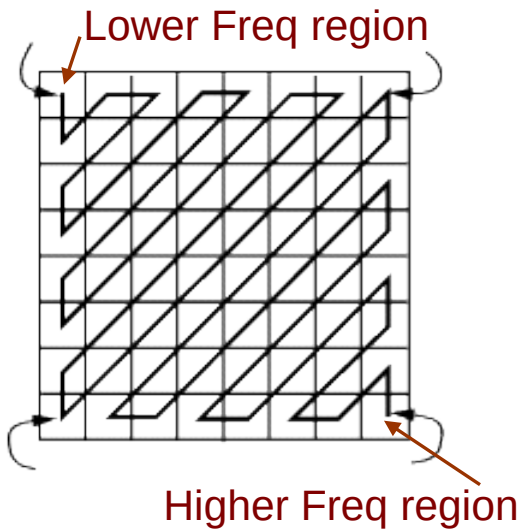
-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Example, T'(0,0) = Round[T(0,0)/Z(0,0)] = Round[-415/16] = -26
- Notice that most entries are now zero.

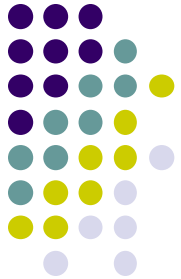
Entropy Encoding



- Similar to monochrone case
- By ordering the coeffs in zig-zag sequence (most trailing coeffs will be zero)
- Use RLE to encode the resulting sequence
- Use Huffman or Arithmetic coding to minimize the coding sequence
- Store the encoded image as JPEG File

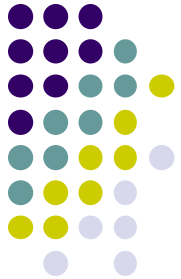


TOPICS



- The Color Models (Chapter 4)
- Color JPEG Standard (S 9.1.1)
- JPEG Modes (S 9.1.2)
- New Image Compression Proposals

Four Commonly Used JPEG Modes



- Sequential Mode:
  - The default JPEG mode, implicitly assumed in the discussions so far
  - Each graylevel image or color image component is encoded in a single left-to-right, top-to-bottom scan.
- Progressive Mode
- Hierarchical Mode
- Lossless Mode: Employs only differential coding method. Compression ratio is low; rarely used

Progressive Mode -1

- Progressive JPEG delivers low quality versions of the image quickly, followed by higher quality passes.

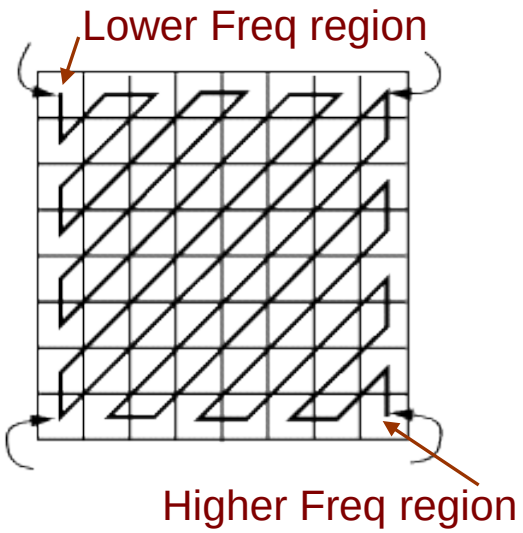
1. **Spectral selection:** Takes advantage of the "spectral" (spatial frequency spectrum) characteristics of the DCT coefficients: higher AC components provide detail information.

Scan 1: Encode DC and first few AC components, e.g., AC1, AC2.

Scan 2: Encode a few more AC components, e.g., AC3, AC4, AC5.

...

Scan k: Encode the last few ACs, e.g., AC61, AC62, AC63.



Progressive Mode -2

2. Successive approximation: Instead of gradually encoding spectral bands, all DCT coefficients are encoded simultaneously but with their most significant bits (MSBs) first.

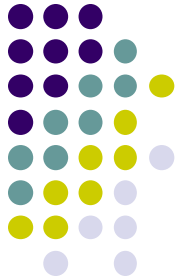
Scan 1: Encode the first few MSBs, e.g., Bits 7, 6, 5, 4.

Scan 2: Encode a few more less significant bits, e.g., Bit 3.

...

Scan m: Encode the least significant bit (LSB), Bit 0.

Hierarchical Mode -1



- Encoding of an image is done at different resolutions
- Algorithm:
  - Down-sample the original image by a factor of 2 in each dim.
  - Encode reduced image used the standard JPEG compression
  - Decode this compressed image and up-sample it
  - Use this up-sampled image as a prediction of the original image and encode the difference using standard JPEG
- Repeat this algorithm to obtain successively lower resolution images

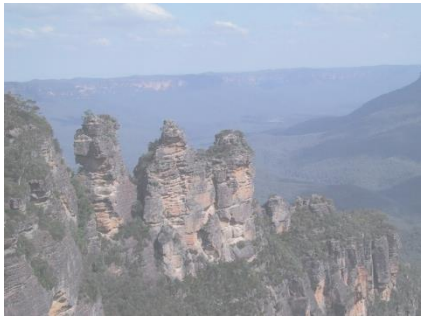
Hierarchical Mode -2

- Derive: Diff($I_0 - I_{-1}$) = $I_0 - I_{-1}$
- Then: Jpeg compress : $J(I_{-1})$

  Jpeg compress : $J(\text{Diff}(I_0 - I_{-1}))$
- $J(I_{-1})$ gives lower resolution image at lower quality, but at much higher compression rate
- $J(\text{Diff}(I_0 - I_{-1}))$ gives the difference
- Together, we can then obtain I_0 from I_{-1} + Diff($I_0 - I_{-1}$)

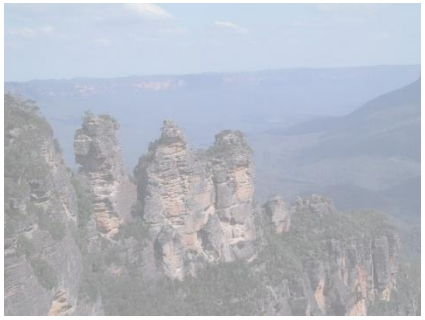
Hierarchical Mode -3

Original Image I_{-1}



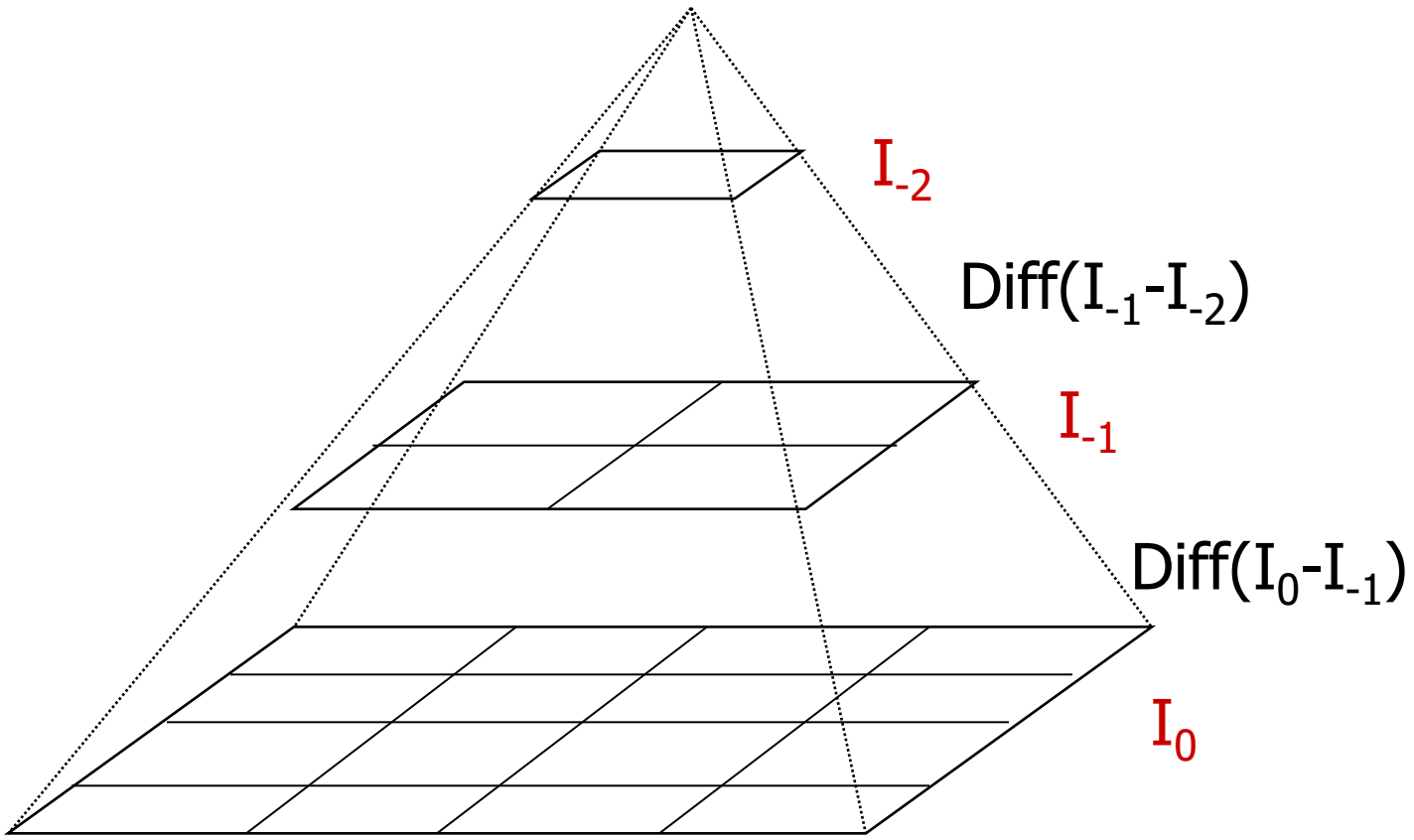
Down-sample
By, say, 2 → $I_{-1/2}$

Un-sample by 2 → I_{-2}

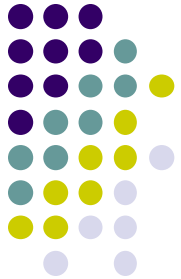


- Repeat the process to derive: Diff($I_{-1} - I_{-2}$) = $I_{-1} - I_{-2}$
- Then: Jpeg compress : J(I_{-2})
  Jpeg compress : J(Diff($I_{-1} - I_{-2}$))
- Together, we can then obtain I_{-1} from I_{-2} + Diff($I_{-1} - I_{-2}$)
- Etc..

Hierarchical Processing -4

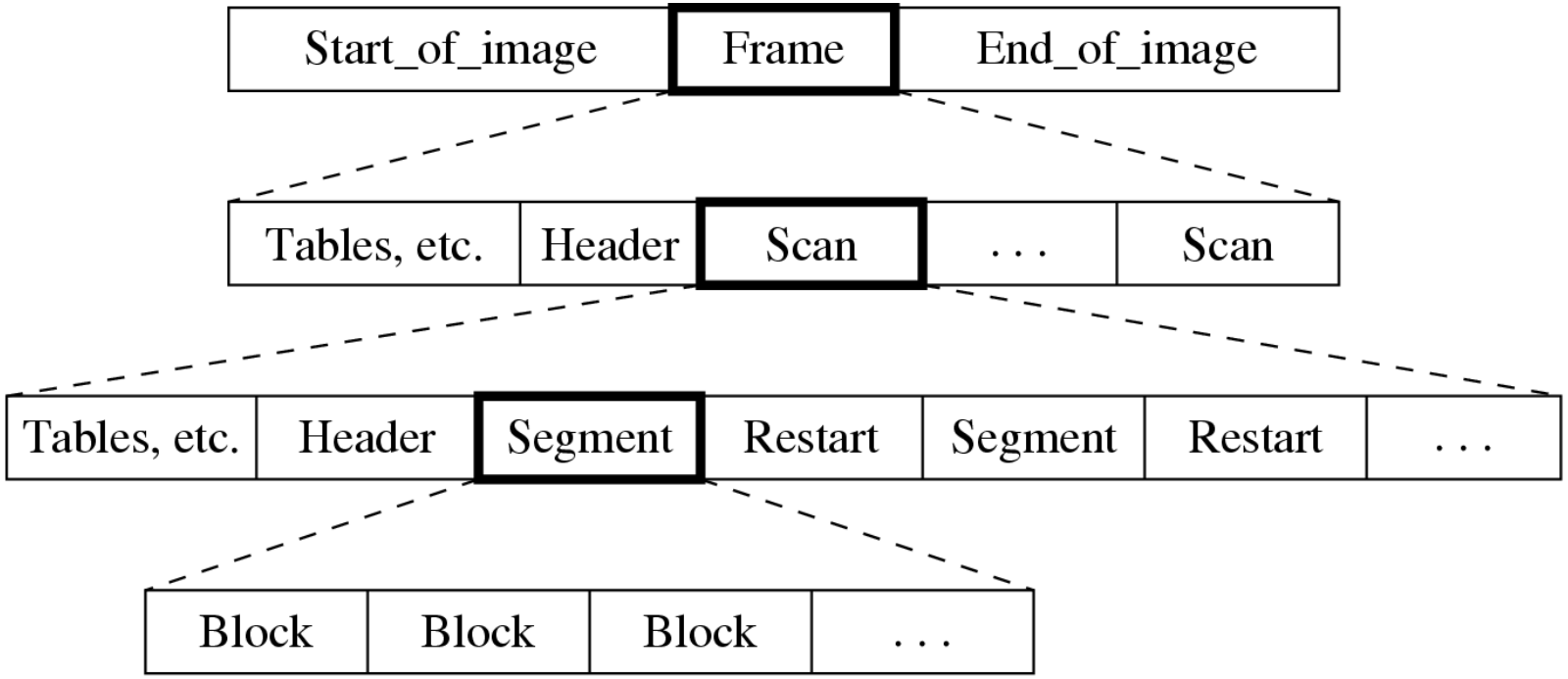


Hierarchical Mode -5

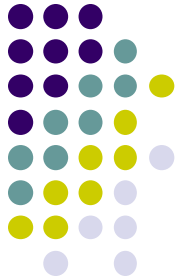


- Encoding of an image is done at successively lower resolutions as shown
- Decoding can be started at the lowest resolution and repeated till the highest resolution is reached
- The content can be transmitted successively, improving quality
- More storage is needed

A Glance at the JPEG Bitstream



JPEG Header



Frame Header:

- Bits per pixel
- Width, height of image
- Number of components
- Unique ID (for each component)
- Horizontal/vertical sampling factors (for each component)
- Quantization table to use (for each component

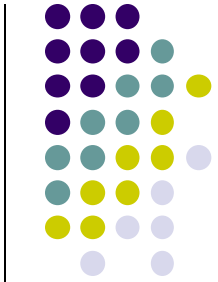
Scan Header:

- Number of components in the scan
- Huffman table for each component

Miscellaneous info:

- comments, user application data

TOPICS



- The Color Models (Chapter 4)
- Color JPEG Standard (S 9.1.1)
- JPEG Modes (S 9.1.2)
- New Image Compression Proposals

JPEG 2000

- A new image coding system that aims to tackle the following problems (http://www.jepg.org/jpeg2000)
  - To offer superior quality at lower bit rates of <0.25 bits/pixel for gray-scale images: artifacts are less visible and there is almost no blocking.
    - JPEG tends to have “blocking and ringing” artifacts
  - Offer superior compression performance for images > 64x64K without tiling
  - Provides lossless & lossy compression – *Use of DWT (Discrete Wavelet Transform)*
  - It requires larger compression time than JPEG, but allows for more sophisticated progressive downloads

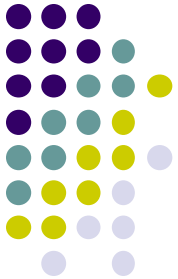
Original Lena image
(256x256 pixels, 24-bit RGB)

JPEG2000 compressed
(compression ratio 43:1)

JPEG compressed
(compression ratio 43:1)

Size of this preview:
133 × 599 pixels

Some History & Future Trends



1990s

JPEG

Reasons for not widely used
1. Not backward compatible with JPEG.
2. Too memory consuming at that time.

2000s

JPEG 2000 → Motion JPEG 2000

JPEG XR (*Microsoft 2006*)

2010s

WebP (*Google 2010*)

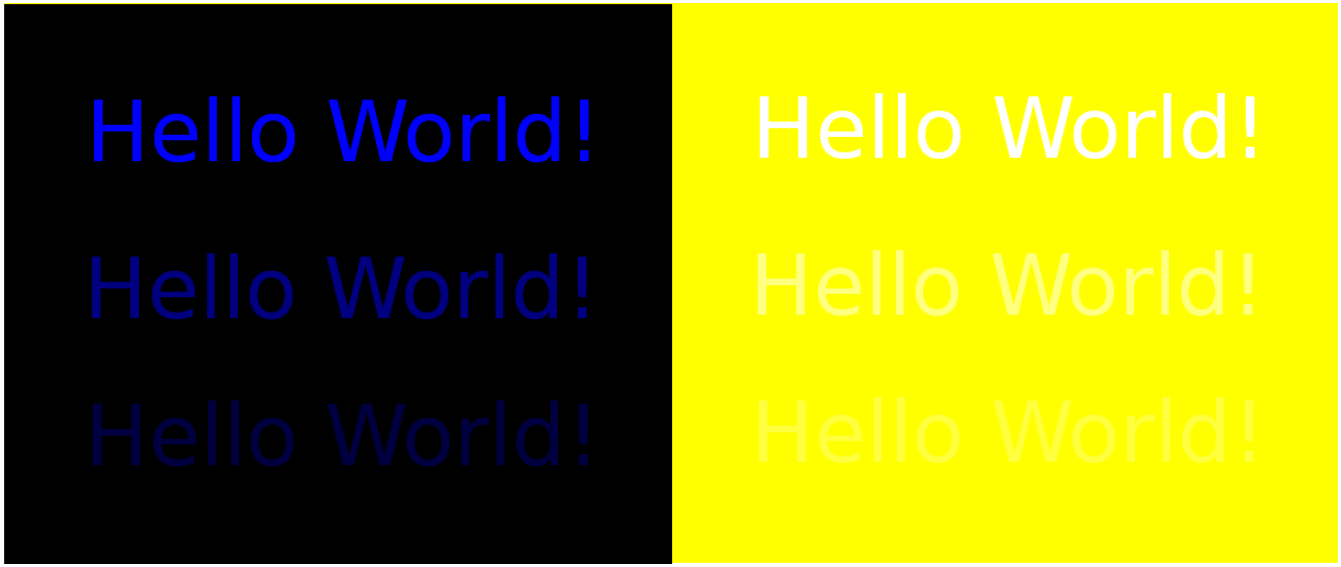
MozJPEG (*Mozilla 2014*)

BPG (*2014*)

Guetzli (*Google 2017*) ← The latest and most advanced proposal

Guetzli

- Its goal is to find the smallest JPEG which cannot be distinguished from the original image by human eyes.
- Intuitions: eliminate information not effective
  - Changes in blue vicinity of yellow can be encoded less precisely.

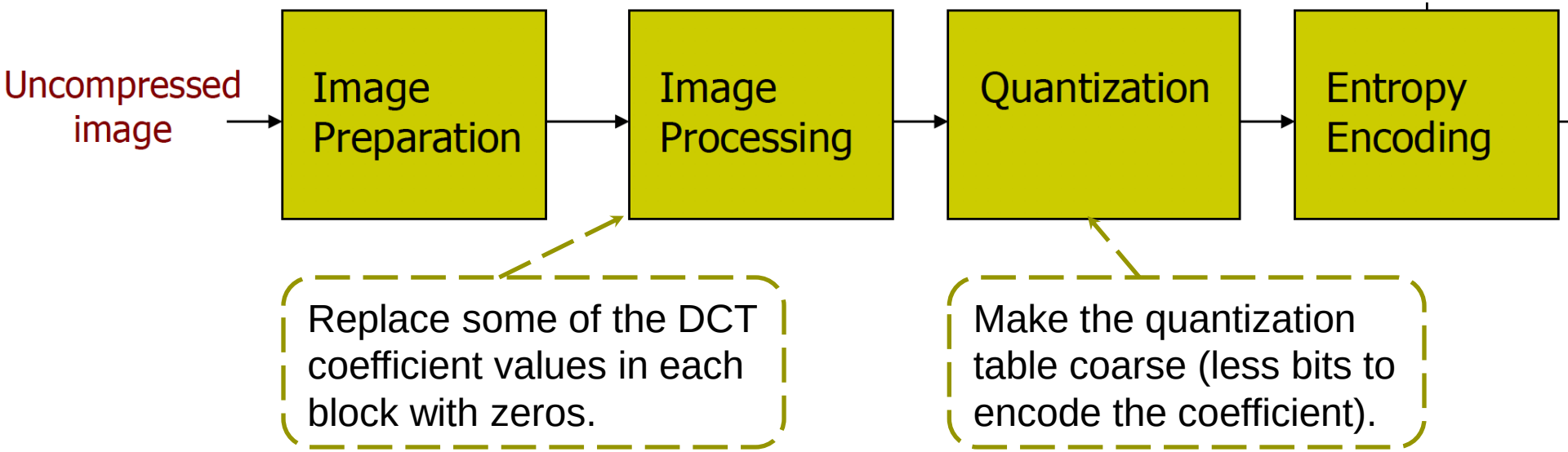


Guetzli

- Its goal is to find the smallest JPEG which cannot be distinguished from the original image by the human eye
- Intuitions: eliminate information not effective
  - Changes in blue vicinity of yellow can be encoded less precisely
  - We can encode areas with large amount of visual noise less precisely

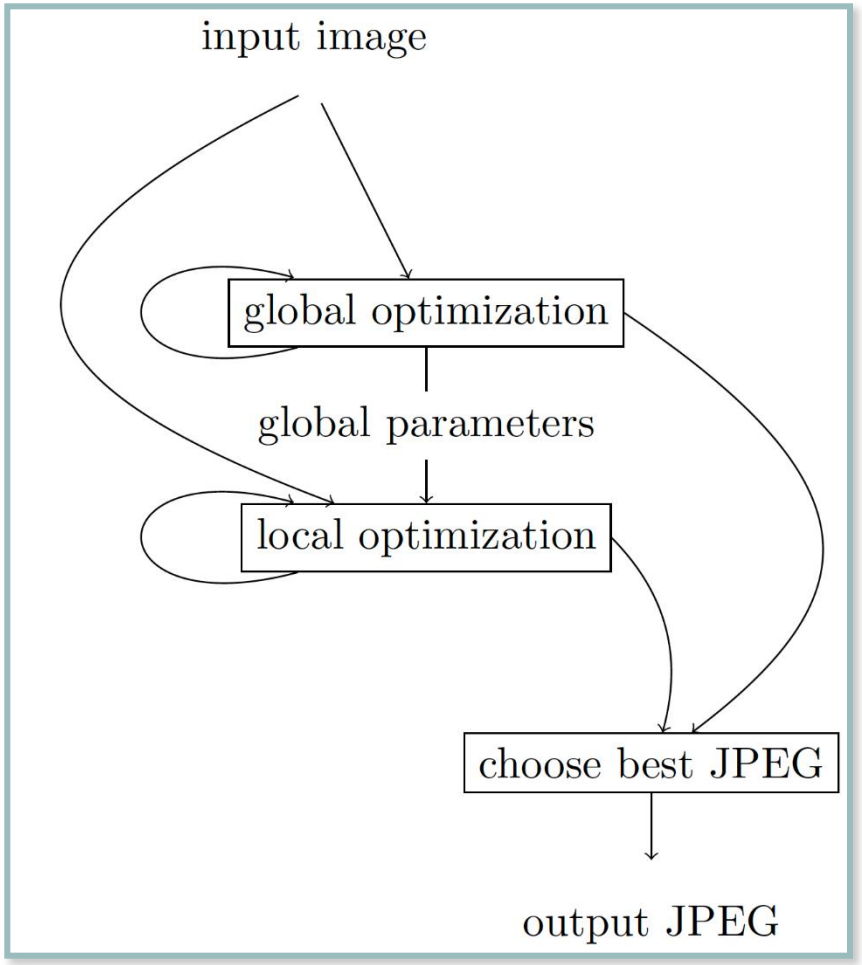
Guetzli

- Its goal is to find the smallest JPEG which cannot be distinguished from the original image by the human eye.
- Intuitions: eliminate information not effective
  - Changes in blue vicinity of yellow can be encoded less precisely
  - We can encode areas with large amount of visual noise less precisely

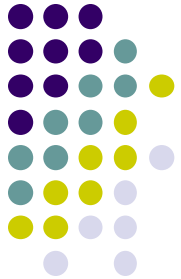


Guetzli

- Guetzli uses an iterative optimization process. In each iteration, Guetzli makes two adjustments, global one (making the quantization table coarse) and local one (replacing coefficients with zeros).

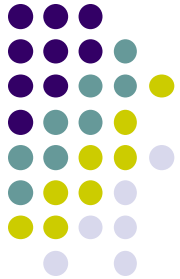


Guetzli: Iterative Optimization -1



- Global adjustment:
  - A global quantization table has 192 values (8 * 8 * 3). Guetzli tries to find **a quantization table** that will produce **psycho-visual distance** (Butteraugli) not larger than **0.97** times the desired maximal distance, when no other distortions are applied.
  - **Psycho-visual distance** is a metric that quantifies the distance between two images by edge structure distortion, based on the assumption that the human visual system is highly adapted to extract structural information from the viewing field.
  - **0.97** was chosen experimentally by Google.
  - **A quantization table** is found from a set of quantization tables experimentally predefined instead of exhaustively searched since it is infeasible to do exhaustive search in that space.

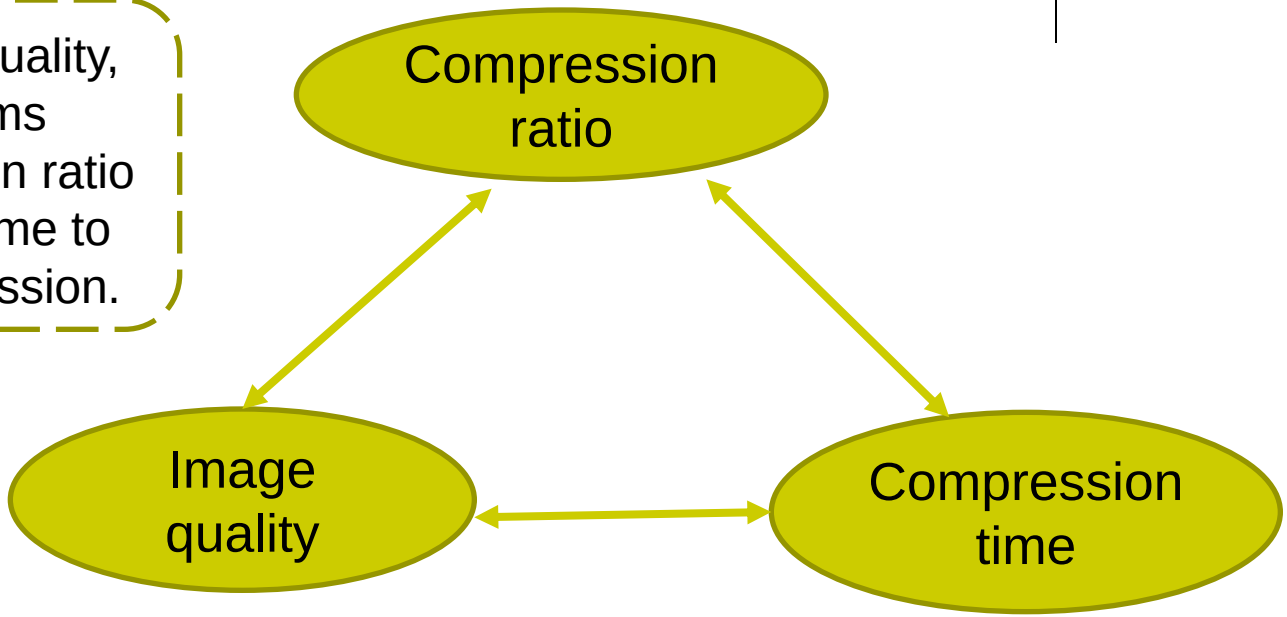
Guetzli: Iterative Optimization -2



- Local adjustment:
  - First calculate an importance value for each coefficient by the variance of psycho-visual distance (variance ↑ importance ↓).
  - Zero-out coefficients with small importance value and make the psycho-visual distance as close as possible to the threshold.

All Tradeoff

Under same image quality, one algorithm performs better on compression ratio usually takes more time to generate the compression.

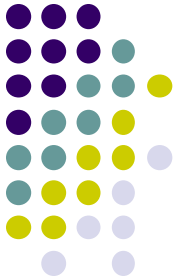


- No perfect compression algorithm
- Internet companies prefer to reduce image size under quality constraint
  - User experience > network transformation & storage > server computation

References on Guetzli

- Zhang, M., & Mou, X. (2008, October). A psychovisual image quality metric based on multi-scale structure similarity. In Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on (pp. 381-384). IEEE.
- Alakuijala, J., Obryk, R., Stoliarchuk, O., Szabadka, Z., Vandevenne, L., & Wassenberg, J. (2017). Guetzli: Perceptually Guided JPEG Encoder. arXiv preprint arXiv:1703.04421.

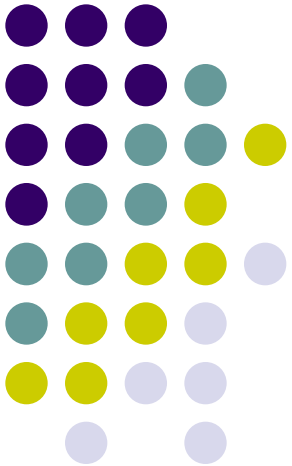
NEXT LESSON



- MPEG Video Standards

Introduction to Media Computing

Lecture 14: Digital Video and MPEG Standards



Lecturer: Dr. Rajiv Ratn Shah, *rajivratn@iiitd.ac.in*

TA: Vaishali Dabral, *vaishali17066@iiitd.ac.in*

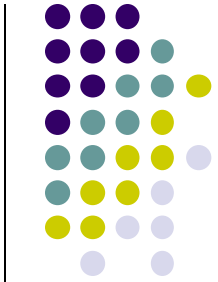
TOPICS

- Digital Video and Its Representation
- MPEG 1 & 2 Standards

References:

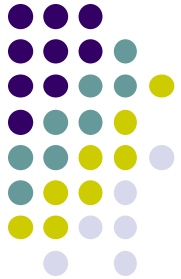
- Li & Drew,: Chapters 5, 10, 11

Introduction



- Motion video (or simply video) is a combination of image and audio
  - It consists of a sequence of still images called frames
  - Frame rate: frames per second (fps)
- If frame rate is fast enough, it will create an illusion of motion
  - The frame rate should be around 30 for perceiving smooth realistic motion
  - Audio is added and synchronized with the apparent movement of images

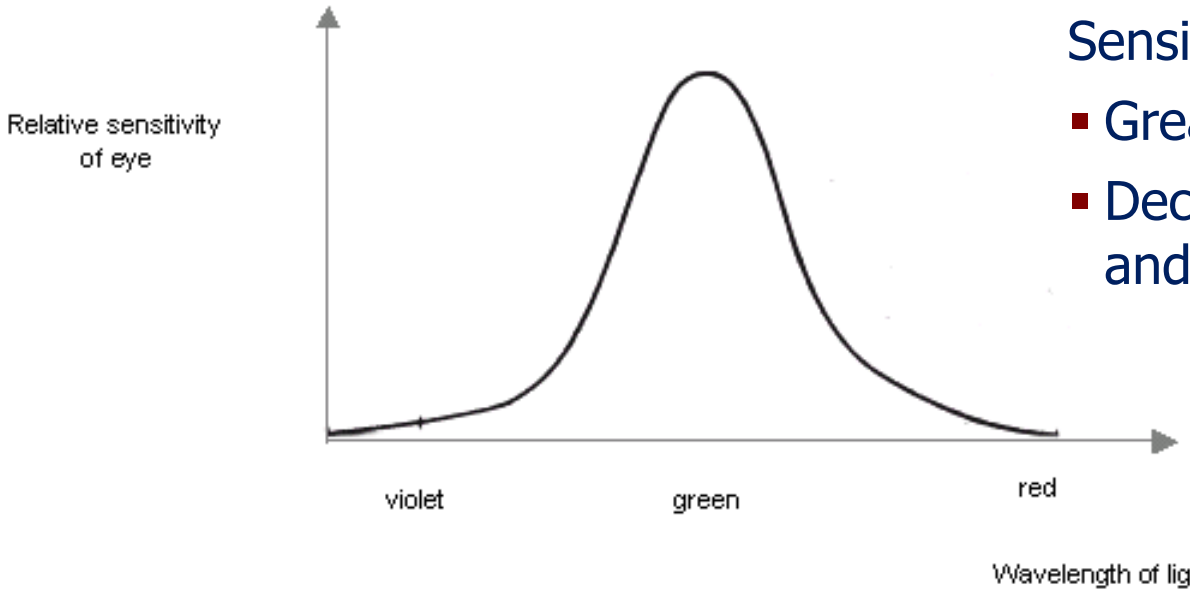
Digital Video



- Many advantages of digital video representation:
  a) Can be stored on digital devices or in memory, ready to be processed and integrated to various multimedia applications.
  b) Direct access is possible, which makes nonlinear video editing achievable as a simple, rather than a complex, task.
  c) Repeated copying does not degrade image quality.
  d) Ease of encryption and better tolerance to channel noise.

Another View of Color Model -1

- According to the theory formulated by Helmholtz the light sensitive cells are of two types – rods and cones
  - The rods provide brightness sensation and thus perceive objects in various shades of grey from black to white
  - The cones that are sensitive to color are broadly divided in three different groups –blue, red and green shades.
  - The human color perception curve



Sensitivity of human eye is:

- Greatest for green-yellow range;
- Decreasing towards both the red and blue ends of the spectrum

Another View of Color Model -2

- Relative intensities of the primary colors for color transmission (e.g. for color television) has been standardized
  - Reference white = of 30% red, 59% green and 11% blue

- Digital video adopts the YC_rC_b format
  - Y (or luminance): $Y = 0.3\ R + 0.59\ G + 0.11\ B$
  - C (or chrominance):

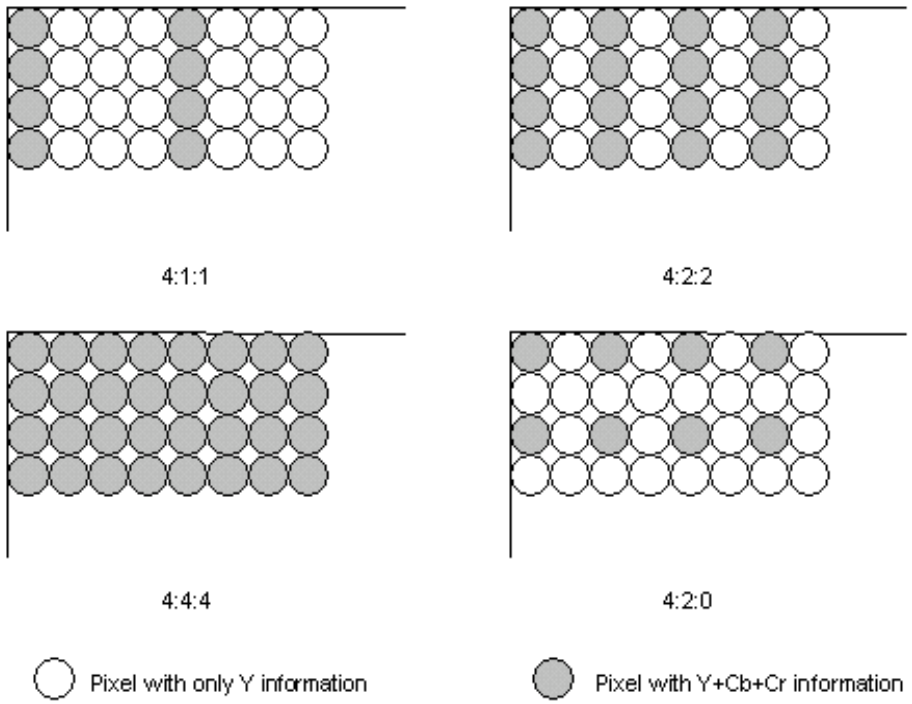
    $C_b = B - Y$

    $C_r = R - Y$

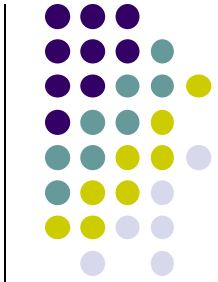
- Since humans see color with much less spatial resolution than they see luminance (or black and white), it makes sense to "decimate" the chrominance signal

Video Representation

- Various schemes of chroma sub-sampling
  - The 4:4:4 scheme implies no chroma subsampling
  - In 4:2:2 scheme, for every 4 units (pixels) of Y info, only 2 units each of Cb and Cr will be transmitted
  - In 4:1:1 scheme, the chroma is sub-sampled at one fourth the horizontal frequency as luma
  - In 4:2:0 scheme performs both horizontal & vertical sub-sampling. I.e. for every 4 pixels of Y information there are 2 pixels of C info both along a row as well along a column

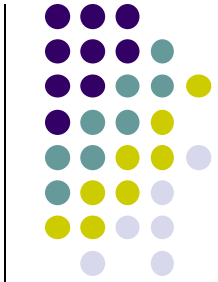


What is the Bottleneck for Digital Video?



- Let's look at the raw data rates for digital audio and video
- CD quality digital audio
  - 44 kHz sampling rate x 16 bits/sample
  - Approximately **700 kbps**
- High definition video
  - 1,280 pels x 720 lines luma
  - 640 pels x 360 lines chroma x 2 channels
  - 60 frames/s x 8 bits/pel/channel
  - Approximately **663.5 Mbps**
  - (from the GA-HDTV proposal)
- Approximately 1,000 times the audio size ➔ a picture is worth a thousand words! (Confucius, 500 BC)
- Compression is the KEY

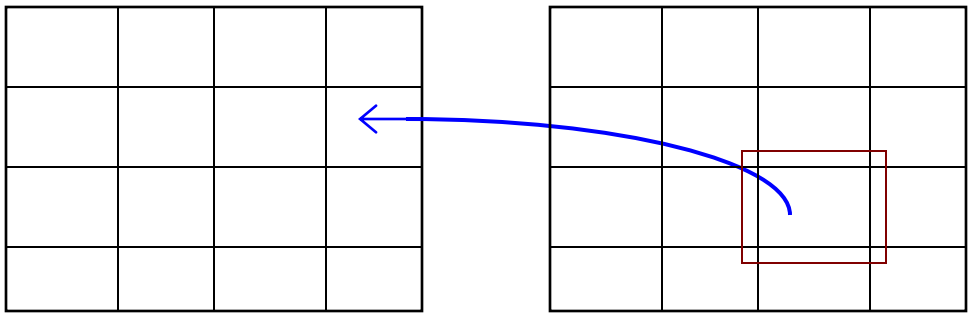
General Video Compression



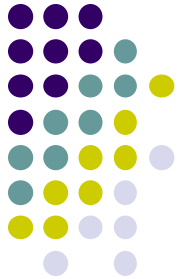
- Types of redundancy in images – a review
  - Spatial redundancy – reduce by intra-frame coding
  - Perceptual redundancy – reduce by quantization and differential coding of color components
  - Coding redundancy – reduce by Run-length and Huffman coding
- What is the addition type of Redundancy in video?
  - Consecutive frames carry similar contents → Temporal redundancy
- Key is in removing temporal redundancy:
  - Needs to consider inter-frame coding
  - Results in most significant reduction

Temporal Redundancy – Basic Idea

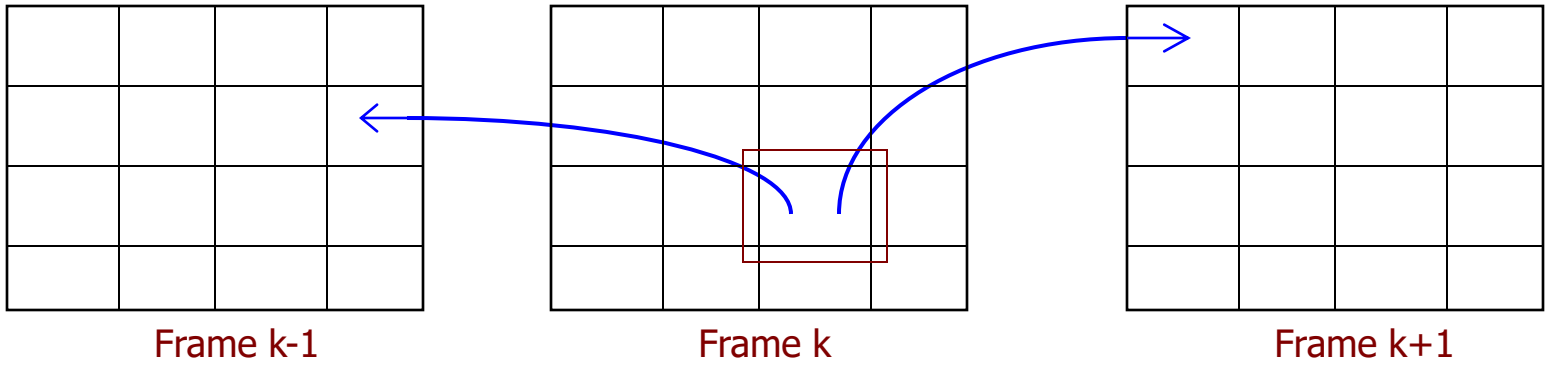
- Without utilizing temporal redundancy
  - Code as JPEG block (or intra coded)
  - Not much saving can be made
- Principle of temporal redundancy removal:
  - Consecutive frames carry similar but slightly varying content
  - Hence, predict contents of current frames from previous or subsequent
- How to do it?
  - Hard to do it at the whole frame level
  - Simplest is to divide frame content into blocks (non-overlapping)
  - Try to find identical or similar blocks in the previous frames



Temporal Redundancy -2



- With the use of temporal redundancy contents
  - If identical block can be found, need only to code {dx, dy}
    ➔ max saving
  - Otherwise, code as difference between current & predicted block, or {dx, dy, δB}, where $\delta B = B_k(x,y) - B_{k-1}(x,y)$

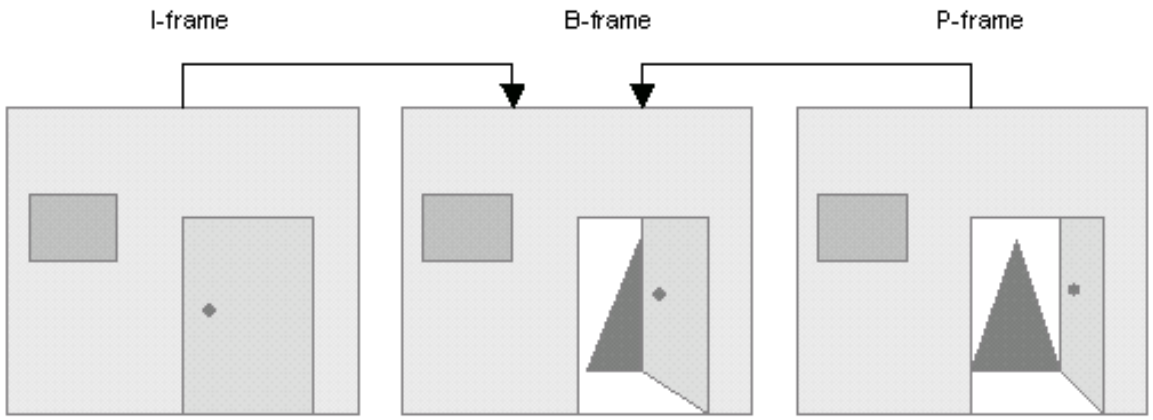


- Even better if we can consider both a previous and a following frame

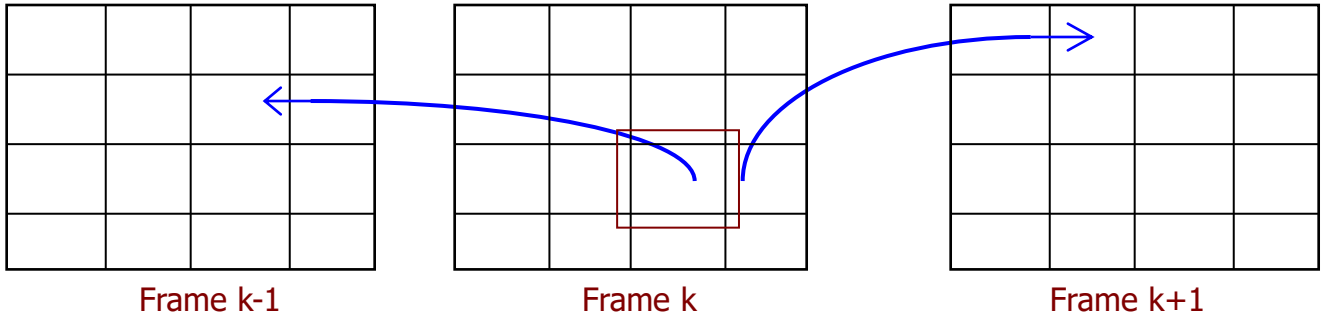
  Better chance of finding a similar block – **WHY??**

Temporal Redundancy -3

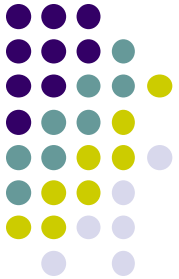
- Reason that we can do better with bi-directional search
  Answer: Able to handle occlusion etc..



- Questions are: How to search, where to search etc.
  Ensure efficiency and accuracy – a Search problem



Motion Compensation Model

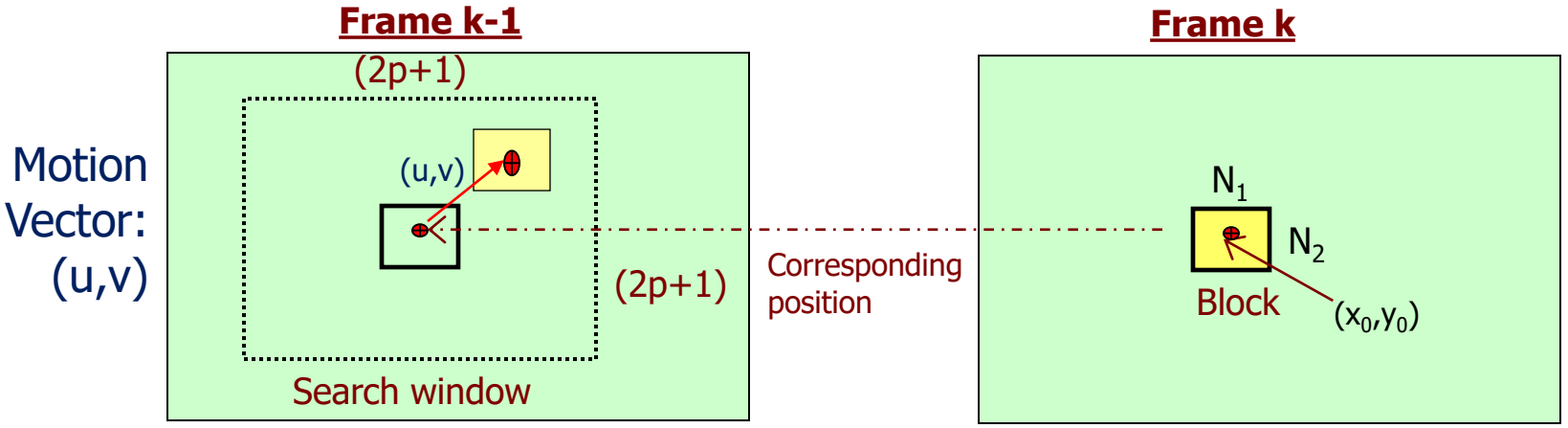


- Block-based Motion Compensation
  - Principle: predict contents of current frames (at block level) from previous or subsequent frames
  - MPEG uses Macroblocks, of 16x16 pixel (or 2x2 DCT) blocks as motion compensation units
  - Motion information comprises the direction and amplitude of displacement of the contents, {dx, dy, δB}
  - Consider only 2D translation & non-overlapping model
- Advantages Block-based Motion Compensation
  - Low overhead - needs only one motion vector per block
  - Availability of low-cost VLSI implementation
- Disadvantages
  - Fails for zoom, rotation motion and under local deformation
  - Discontinuity at block boundaries
  - Serious blocking artifacts, especially at low bit-rate

Idea of Block Translation Model

- Given a block to be encoded in Frame k
  - Block: size N_1 x N_2 centered at (x_0, y_0) at Frame k
  - Assume motion changes moderately, we limit search space in Frame (k-1) to range of [-p, p]

    → Search window of (2p + 1) x (2p + 1) centered at (x_0, y_0)

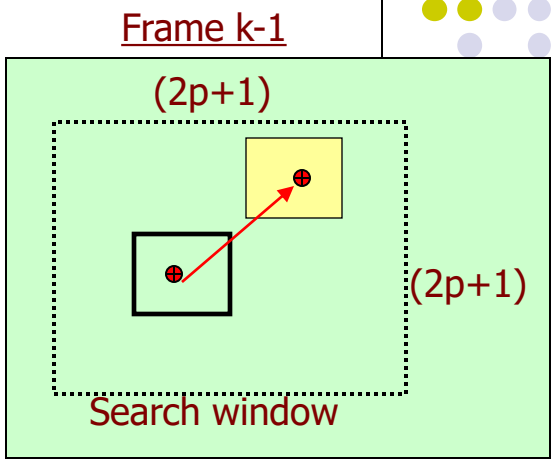


- Aim is to look for a best matched block in Frame (k-1) s.t.:

$S_k(n_1, n_2) \approx S_{k-1}(x_0+u, y_0+v)$ or $Er = \min_{x \in V} \sum_{\bar{x} \in M} Diff[s_k(\bar{x}) - s_{k-1}(\bar{x} + \bar{v})]$

Search Procedures

- Block matching algorithms differ in:
  - Matching criteria (max correlation, min errors etc.)
  - Search strategy – usually employs some variants of binary search



- The matching criterion for macroblocks:
  - Minimum mean square error (MSE):

$$MSE(d_1,d_2)=\frac{1}{N_1N_2}\sum_{(n_1,n_2)\in\aleph}[s_k(n_1,n_2)-s_{k-1}(n_1+v_1,n_2+v_2)]^2$$

  - Minimum mean absolute difference (MAD):

$$MAD(d_1,d_2)=\frac{1}{N_1N_2}\sum_{(n_1,n_2)\in\aleph}|s_k(n_1,n_2)-s_{k-1}(n_1+v_1,n_2+v_2)|$$

ERROR=

5	6
1	3

-

4	4
1	1

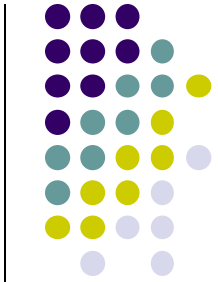
=

1	2
0	2

MAD = 5/4

MSE = 9/4

Full Search

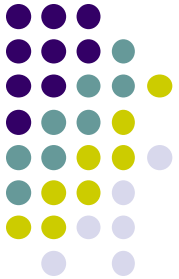


- Full search: calls for the evaluation of the matching criterion at (2p+1) x (2p+1) distinct points for each block
  - Complexity: $(2p+1) \times (2p+1) \times N^2 \times 3 = O(p^2 N^2)$
  - Given an image of resolution 720 x 480 at 30 fps:

    $(2p+1) \times (2p+1) \times N^2 \times 3 \times (720 \times 480) / (N \times N) \times 30 = 29.89 \times 10^9$

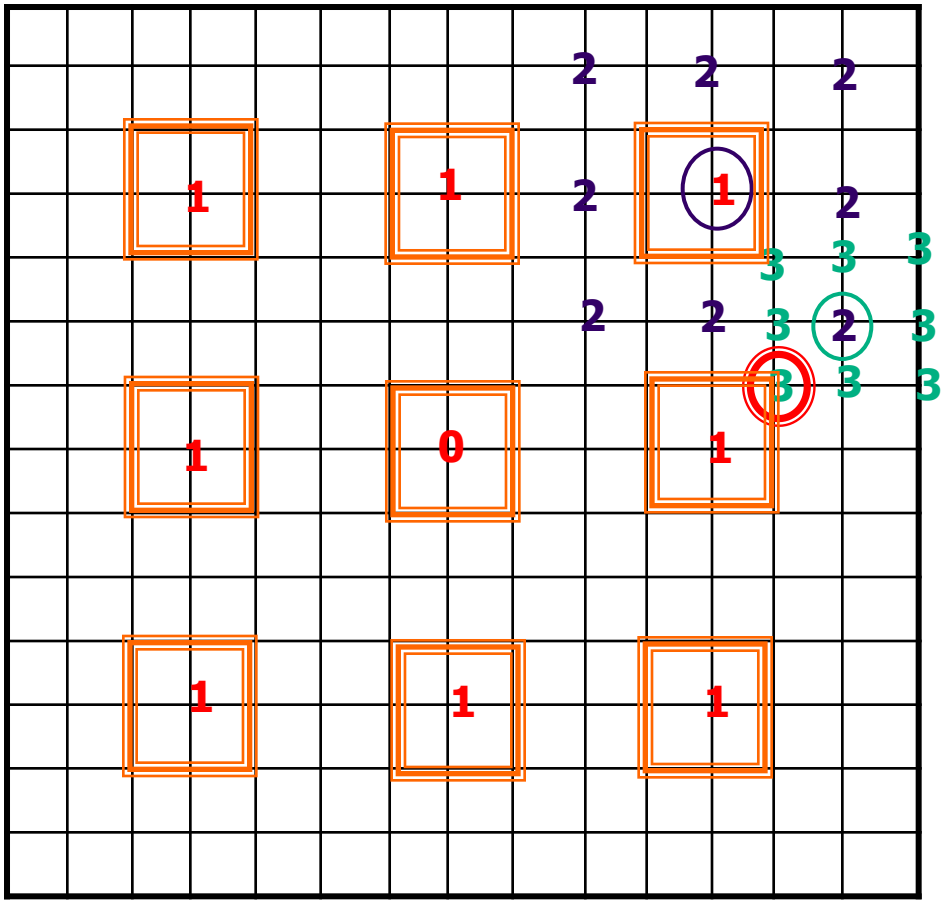
- We can reduce the search complexity by using different search strategies
  - 2D Logarithmic Search
  - Hierarchical Search

2D Log Search (or 3-Step Search)



- Search in selected location to minimize search efforts
  - Criteria of match: best MSE or MAD match

- Let p = 7
  - Step 1: search at locations "0" & "1's"
  - Step 2: search at locations "2's"
  - Step 3: search at locations "3's" at one pixel resolution
  - An logarithmic search strategy



- WHY it works? Assume motion changes slowly & smoothly

2D Log Search (3-Step Search) -2

- Search Complexity:
  - Compare only 8 x ([$\log_2 p$] + 1) microblocks,

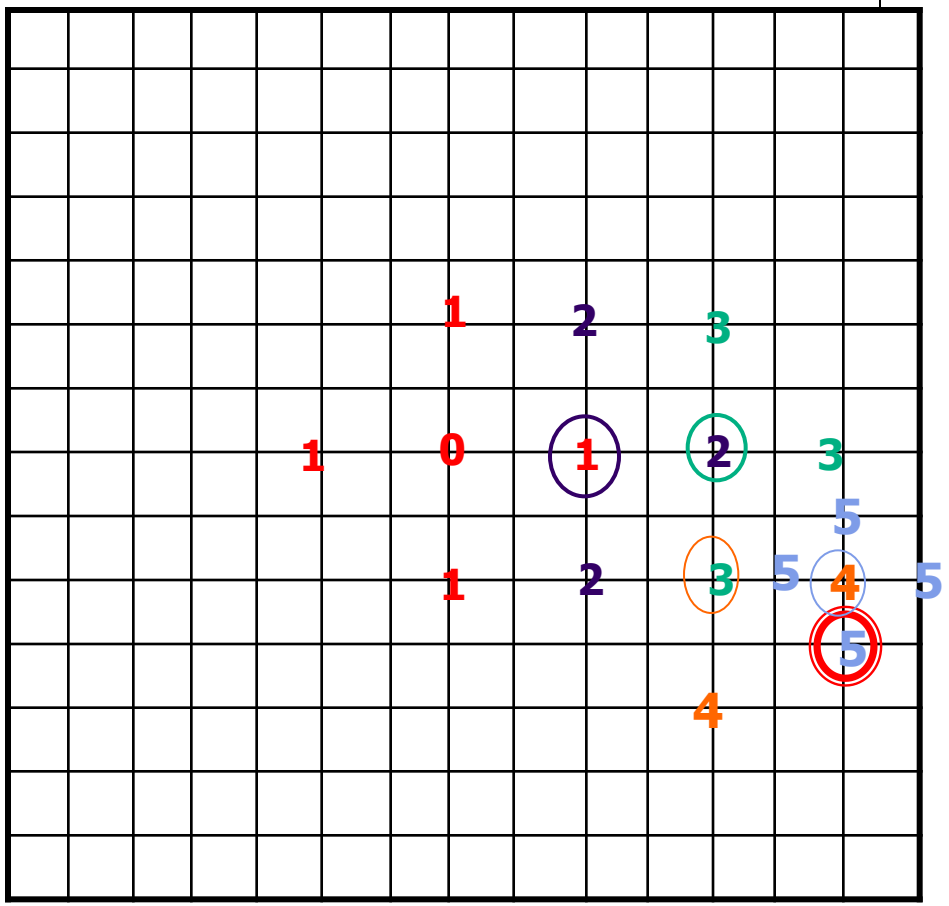
    since the comparison that yields the least MAD from previous iteration can be re-used.
  - Hence the complexity is: $O(\log p \times N^2)$
  - Given an image of resolution 720 x 480 at 30 fps:

$$
\begin{aligned}
OPS\_per\_second &= \left(8\cdot(\log_2 p+1)+1\right)\cdot N^2\cdot 3\cdot\frac{720\times 480}{N\cdot N}\cdot 30 \\
&= \left(8\cdot\log_2 15+9\right)\times 16^2\times 3\times\frac{720\times 480}{16\times 16}\times 30 \\
&\approx 1.25\times 10^9
\end{aligned}
$$

2D Log Search (Cross Search)

- Another logarithmic search strategy
  - o Set P=7

- Step 1: search at locations "0" & "1's"
- Step 2: search at locations "2's"
- Step 3: search at locations "3's"
- Step 4: Search "4's" -corner
- Step 5: Search at "5's" at the corners - FINAL

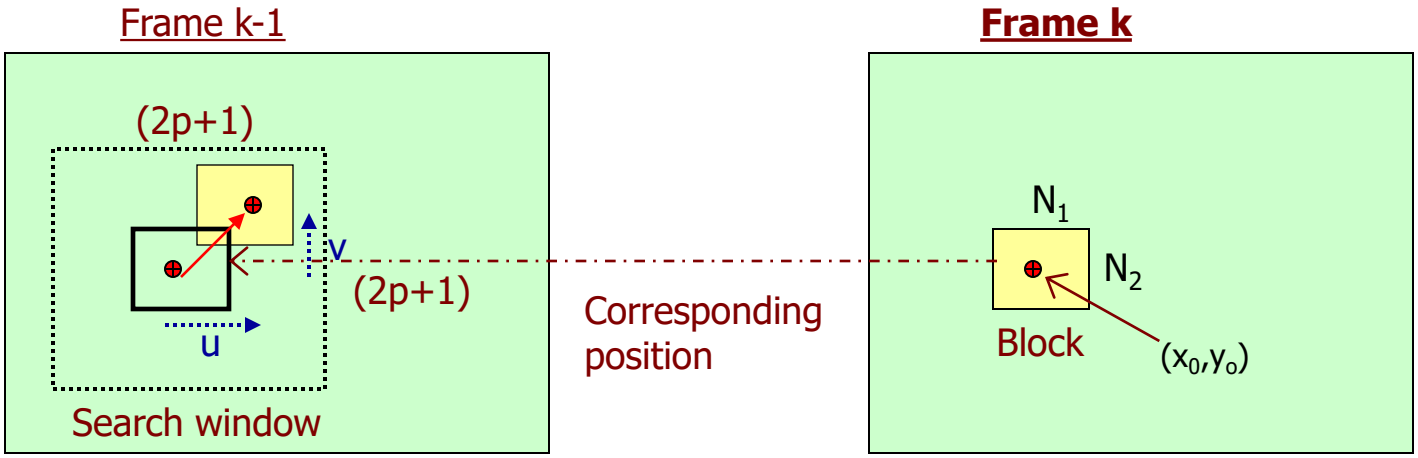


Search will terminate if best match is at center of cross or corner of search window

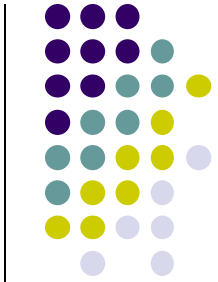
Search Results -1

- Search results – we need to encode 2 pieces of info:
  - Best match: $S_k(x_0, y_0)$ and $S_{k-1}(x_0+u, y_0+v)$
  - Matching Error:

    $D(x_0, y_0) = \Sigma\,\Sigma\,\text{Diff}\{S_k(x_0, y_0) - S_{k-1}(x_0+u, y_0+v)\}$
  - Motion vector: $\underline{m_v} = (u, v)$
  - Hence we need only to encode: $\underline{m_v}$ and $D(x_0, y_0)$ in frame k



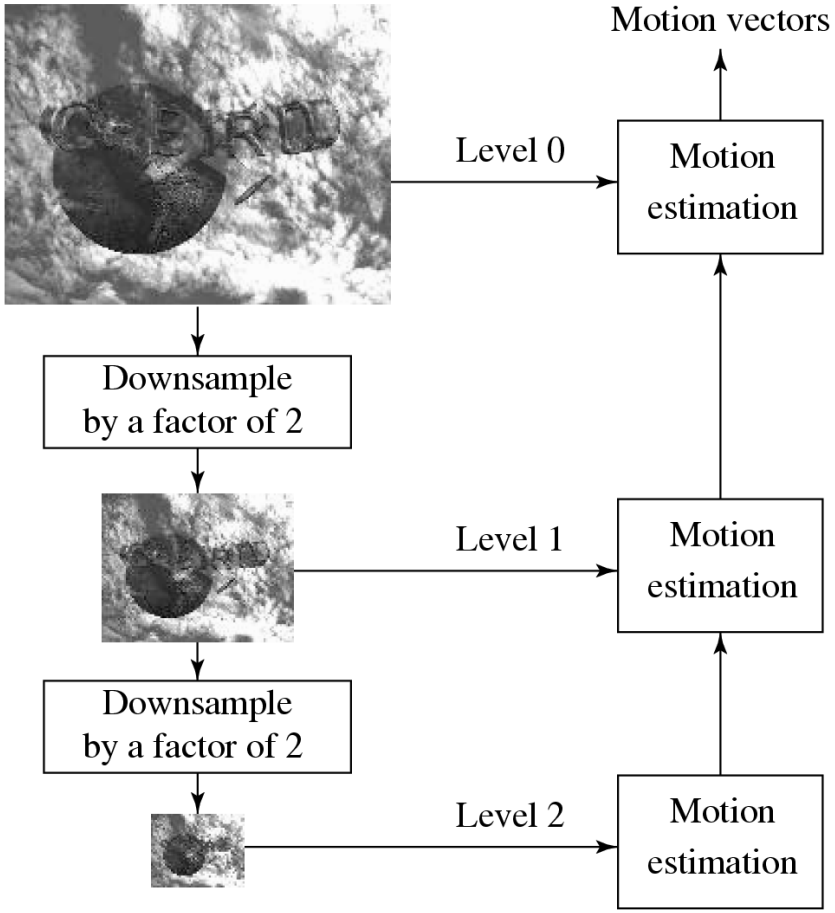
Search Results -2



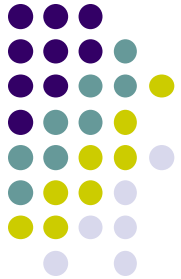
- Depending on magnitudes of $\underline{m}_v$ & error= $D(x_0, y_0)$, we have different encoding scheme:
- If error<σ_1, set $D(x_0, y_0)$=null
  → No point in storing small error vectors
- If $D(x_0, y_0)$=null && $\underline{m}_v$=null, code the block as:
  → skipped macroblock
- If error > σ_3 && $\underline{m}_v$=null, then the block is:
  → intra-coded macroblock (i.e. no motion vector)
- Otherwise, code the block using $D(x_0, y_0)$ and $\underline{m}_v$
- Note: $D(x_0, y_0)$, is coded as normal JPEG image
  - Except that it uses a different quantization matrix -- WHY

Hierarchical Search -1

- Hierarchical (multi-resolution) approach:
  - in which initial estimation of the motion vector can be obtained from images with a significantly reduced resolution
- A 3-level Search:
  - Level 0: original image
  - Levels 1 and 2: obtained by down-sampling from the previous levels by a factor of 2
  - The initial search is conducted at Level 2
  - NOTE: *p* can also be proportionally reduced in down-sampled image



Hierarchical Search -2



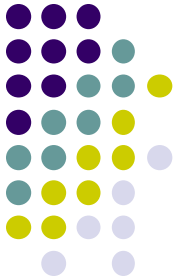
- Given the estimated motion vector (u^k, v^k) at Level k, a 3 x 3 neighborhood centered at $(2 \cdot u^k, 2 \cdot v^k)$ at Level $k-1$ is searched for the refined motion vector.
- The refinement is such that at Level $k-1$ the motion vector (u^{k-1}, v^{k-1}) satisfies:

$$(2u^k - 1 \le u^{k-1} \le 2u^k + 1; \quad 2v^k - 1 \le v^{k-1} \le 2v^k + 1)$$

- Let (x^k_0, y^k_0) denote the center of the macroblock at Level k in the Target frame. The procedure for hierarchical motion vector search for the macroblock centered at (x^0_0, y^0_0) in the Target frame can be outlined as follows:

Hierarchical Search -3

Algorithm: Motion-vector: hierarchical-search



begin

// Get macroblock center position at the lowest resolution Level k

$x^k_0 = x^0_0 / 2^k$; $y^k_0 = y^0_0 / 2^k$;

Use Sequential (or 2D Logarithmic) search method to get initial estimated $\mathbf{MV}(u^k, v^k)$ at Level k;

while last ≠ TRUE

{

Find one of the nine macroblocks that yields minimum MAD at Level $k-1$ centered at:

($2(x^k_0+u^k) - 1 \le x \le 2(x^k_0+u^k) + 1$; $2(y^k_0+v^k) - 1 \le y \le 2(y^k_0+v^k) + 1$);

if $k = 1$ then last = TRUE;

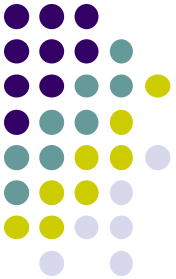
$k = k - 1$;

Assign $(x^k_0; y^k_0)$ and (u^k, v^k) with the new center location and MV;

}

end

Comparison of Computational Cost of Motion Vector Search based on examples

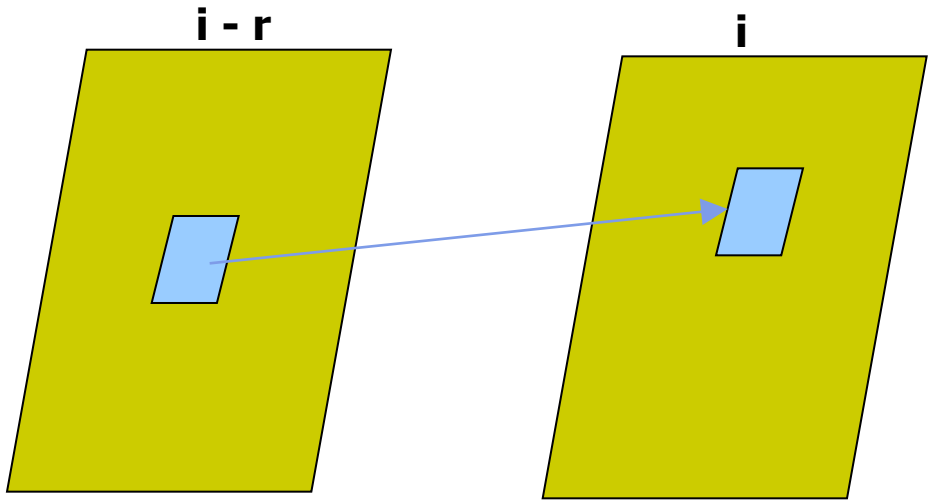


Search Method	OPS_per_second for 720×480 at 30 fps	
	$p = 15$	$p = 7$
Sequential search	29.89×10^9	7.00×10^9
2D Logarithmic search	1.25×10^9	0.78×10^9
3-level Hierarchical search	0.51×10^9	0.40×10^9

Motion Compensation -1

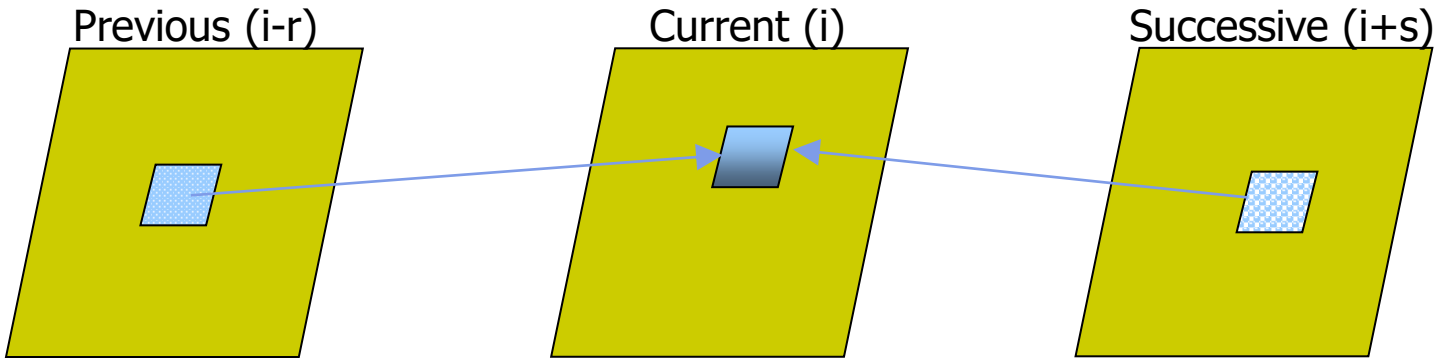
Types of motion compensation on macroblocks:

- Predictive Coding (**P-frames)**:
  - Generate contents of frame k based on contents of a previous frame
  - Example: Frame i is coded based on frame i-r



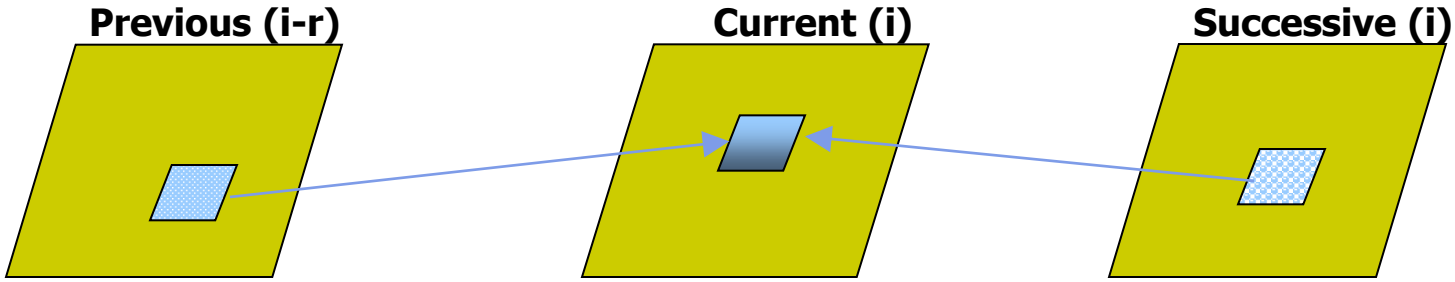
Motion Compensation -2

- Interpolative Coding (**B-Frame**):
  - Contents of a frame is generated based on both a previous and a successive frame
  - (e.g.) Frame i can be coded based on both frames i-r & i+s



- As stated, we need interpolative coding because..
  - Better compression ratio
  - Deals properly with uncovered areas
  - Has better statistical properties (effect of noise is reduced)
  - Allows decoupling between prediction & coding (no error propagation)

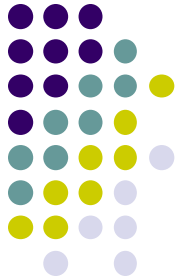
Motion Compensation -3
General Case



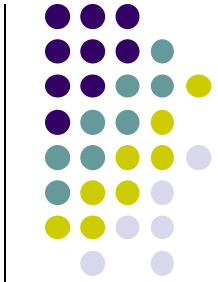
- Predict modes for Macroblock in B-frame
  - Intra: $\Pr edictor: \hat{I}_0(\bar{x}) = null; \quad Error = I_0(\bar{x}) - \hat{I}_0(\bar{x})$
  - Forward-predicted $\Pr edictor: \hat{I}_0(\bar{x}) = I_{-1}(\bar{x} + \bar{v}_{-1,0})$
  - Backward-predicted $\Pr edictor: \hat{I}_0(\bar{x}) = I_{+1}(\bar{x} + \bar{v}_{+1,0})$
  - Bi-direction (average) $\Pr edictor: \hat{I}_0(\bar{x}) = \frac{1}{2}[I_{-1}(\bar{x} + \bar{v}_{-1,0}) + I_{+1}(\bar{x} + \bar{v}_{+1,0})]$
- Motion vectors:
  - one vector for forward and backward predicted macroblocks
  - two vectors for bi-directionally predicted macroblocks

TOPICS

- Digital Video and Its Representation
- MPEG 1 & 2 Standards



Compression Types



Other considerations for compressions include:

Asymmetric Compression

- Compression process is done only once and at the time of storage
- Examples: Video-on-Demand and News-on-Demand servers

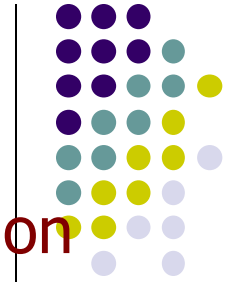
  MPEG is an asymmetric standard

Symmetric Compression

- Equal use of compression and decompression process
- Video generated through video cameras or edited pre-recorded material (like VCR)
- Examples: video conferencing, video telephone, and desktop video publishing (requires edit operations on pre-recorded material)

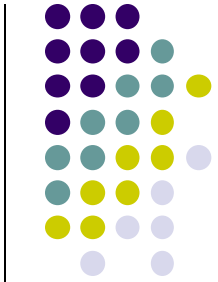
  H.261 is a more symmetric standard

Desired Features



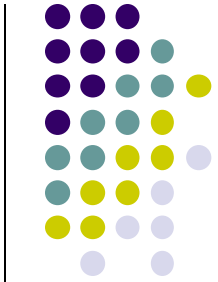
- Balance the needs for random access vs. compression ratio!
- Random Access: Allow viewing from a random point in time
  - Compromise design- *random access points at certain intervals*
  - Typically about 1/2 a second is needed for viewing quality
- Fast-Forward/Rewind : Allow VCR type operations – more demanding form of random access
- Reverse Playback: Playback in reverse mode
- Audio-Visual Synchronization: Video and audio need to be synchronized -- *lip-synchronization*
- Robustness to Errors: Localize errors that might occur during storage or communication
- Coding/Decoding Delay: In video telephony, total system delay should be under 150 ms

Other Desired Features



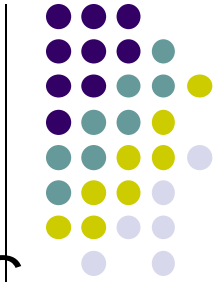
- Asynchronous applications: less strict on compression time. However, limited time for interactive video presentation in store-and-retrieve scenario
- Editability: Need to edit compressed video
- Format Flexibility: Different size of video (depending on window size). Flexibility of the rate of presentation
- Cost Tradeoffs: Implementation in hardware; support encoding in real-time, even if it is in software

Introduction to MPEG



- MPEG: Moving Picture Expert Group
- MPEG-1 was issued in 1992
  - Standard to compress video at a bit-rate of up to 1.5 Mb/sec (Why 1.5 Mbits/sec?)
  - Digital audio signal at the rates of 64, 128 and 192 Kbits/sec
  - Deals with issues related to audio-visual synchronization and multiplexing of multiple compressed audio and video bit streams
- MPEG-1 supports only non-interlaced video. Normally, its picture resolution is:
  - 352 × 240 for NTSC video at 30 fps
  - 352 × 288 for PAL video at 25 fps
  - It uses 4:2:0 chroma subsampling

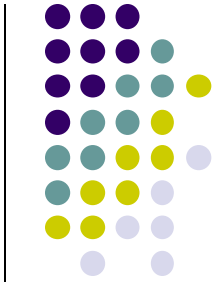
Introduction to MPEG



- MPEG-2 (1994) - higher quality (not lower than NTSC and PAL) with bit rates between 2-10Mb/s.
- VCDs utilize MPEG1
- DVDs utilize MPEG2

- Others: MPEG-4, MPEG-7 and MPEG-21
  - To be discussed in next lecture

Basis of MPEG Encoding



Conflicting Requirements:

- Random access requirements --> pure intra-frame coding
- Higher compression rates ---> inter-frame coding

Basis of MPEG Encoding

- Lossy compression
- Trade off image quality with bit rate according to objective or subjective criteria
- Intraframe coding – similar to JPEG (I-Frame)
- Interframe coding – predicted frame (P-frame) or Bi-directional interpolated frame (B-frame)
- Group of Picture (GOP) – one group of I, P, and B-frames

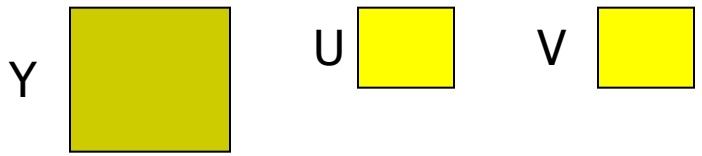
MPEG1 Video Encoding Scheme

STEPS

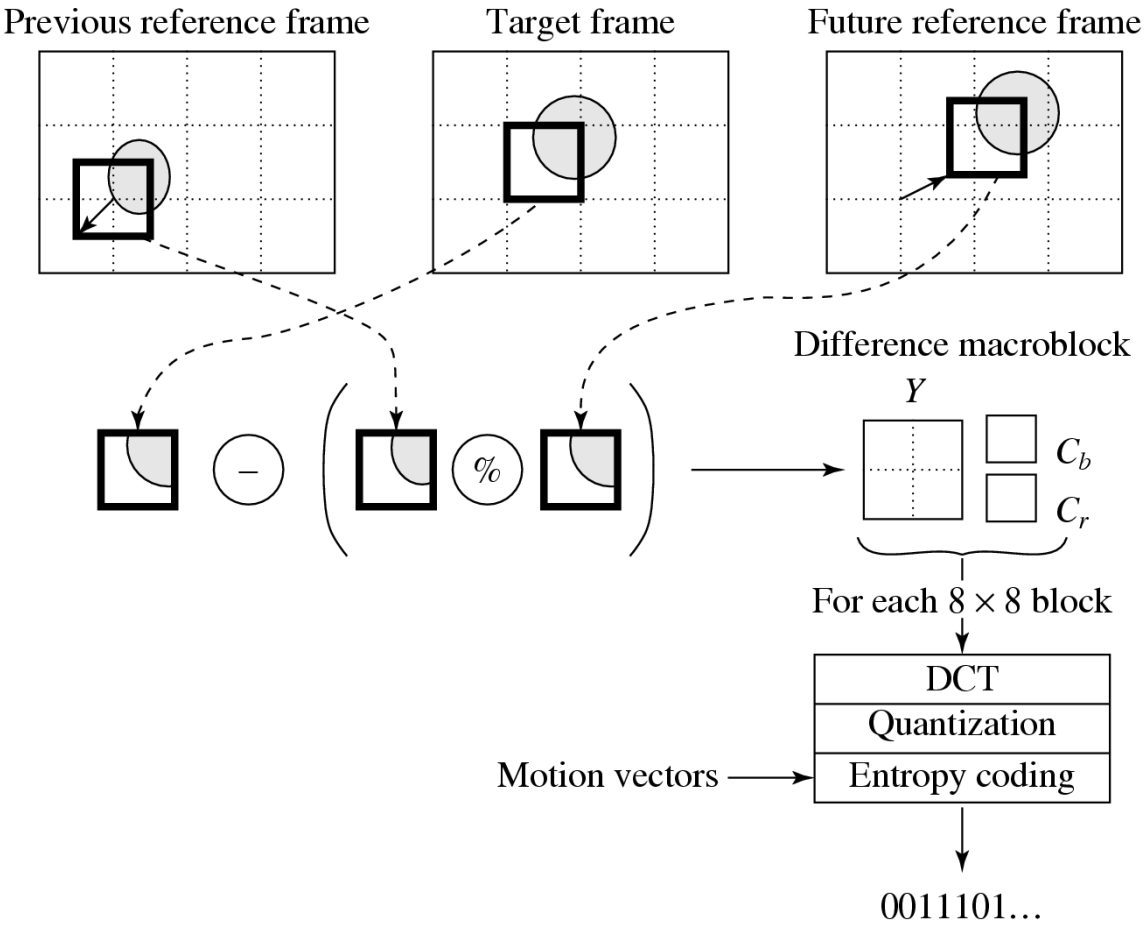
- Partitioning of images into Macroblocks (MB)
- Intraframe coding on one out of every K images - GOP size = k
- Motion estimation on MBs
- Generate (*K*-1) predicted frames
- Encode residual error images

IMAGE FORMAT

- An image is partitioned into Macroblocks of size 16x16
- 1 MB = 4 luminance *(Y)* and 2 chrominance blocks (*U*,*V*)
- The sampling ratio between Y, U and V is 4:1:1

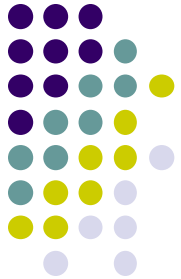


Overview of B-Frame Coding



B-frame Coding Based on Bidirectional Motion Compensation.

Quantization Matrices



8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Default Quantization Table (Q_1) for Intra-Coding

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

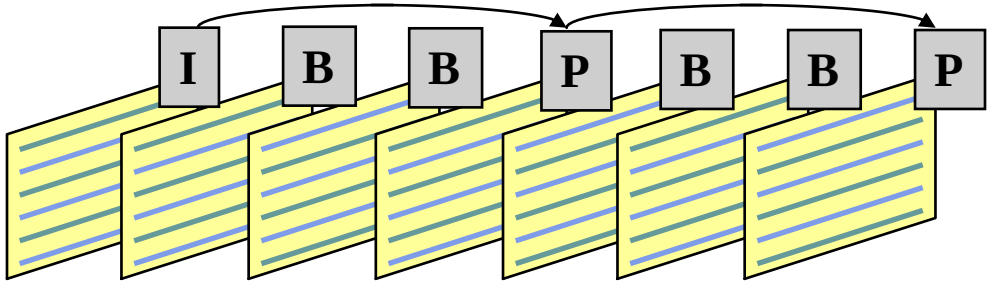
Default Quantization Table (Q_2) for Inter-Coding

Group of Pictures

- Organize frame sequence in terms of Group of Picture (GoP)

  Example: [I B B P B B] [I B B P B B] ….

  - Each GoP is an independent entity
  - I-Frame provides random access point, & as re-synchronization point
  - P and B frames allows for greater compression efficiency



- Other arrangements are possible, for example:

  Example: [I B B P B B P B B P] [I B B P B B P B B P] …

  - Need a I-frame in at least every 1/2 second interval

MPEG1 Encoding Scheme:
Random Access

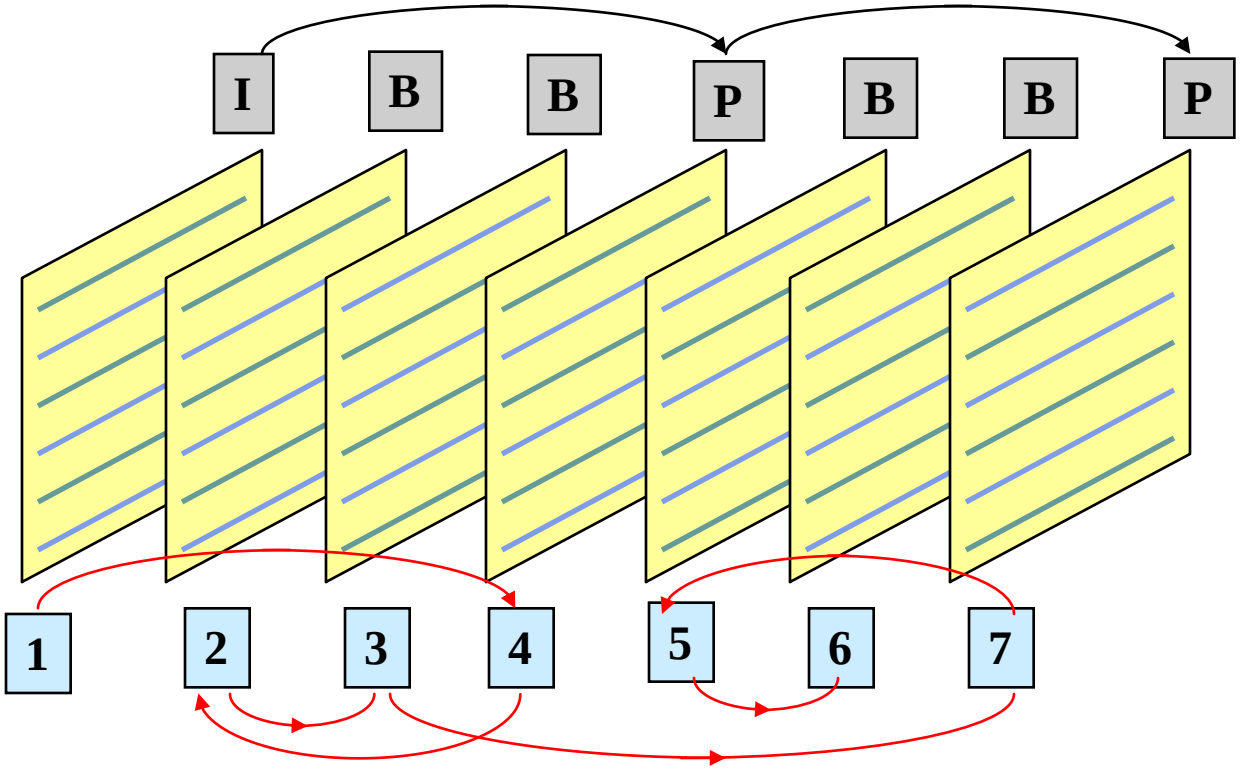


Figure 6

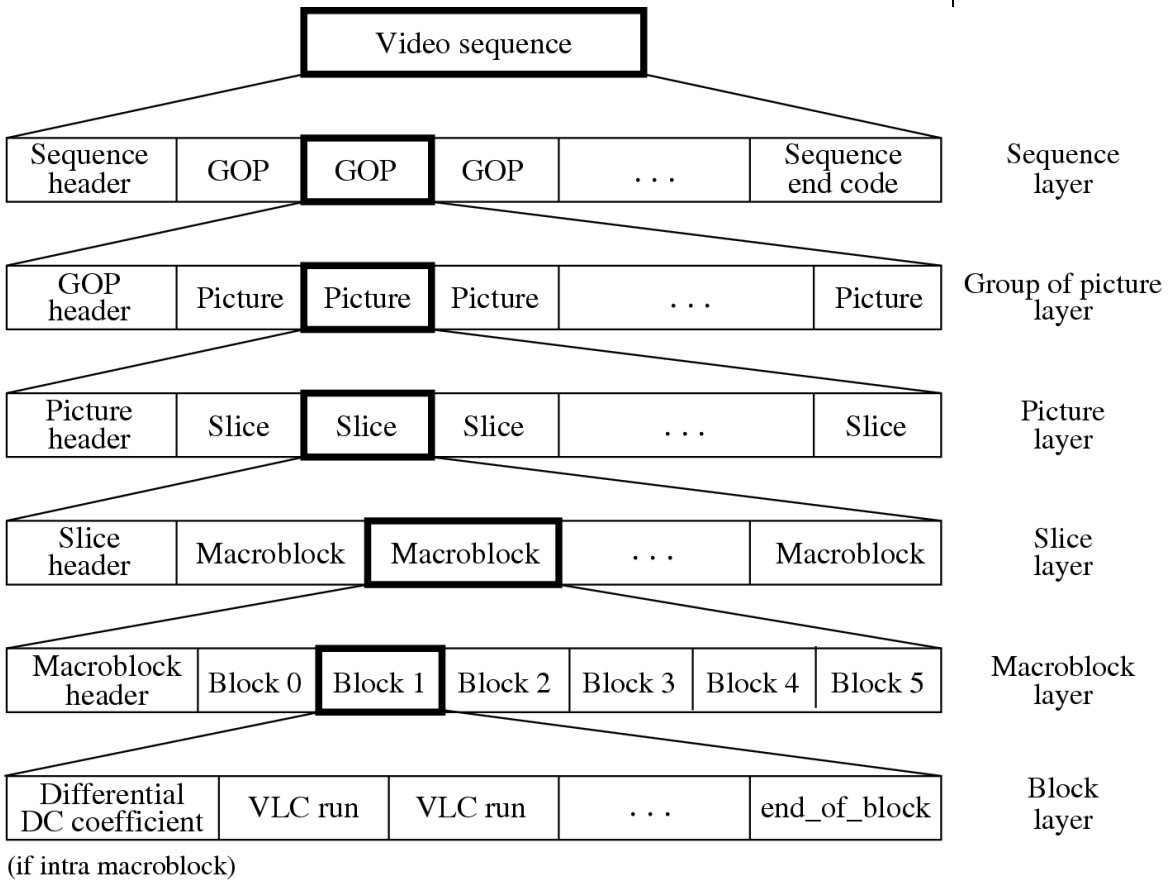
Decoding Order:
1, 4, 2, 3, 7, 5, 6

	Compression	*Random Access*	*Coding Delay*
I Pictures only	**Low**	**Highest**	**Low**
I and P Pictures	**Medium**	**Medium**	**Medium**
I, P and B Pictures	**High**	**Low**	**High**

MPEG Video Bit Stream

Six layers of syntax of MPEG video bit stream

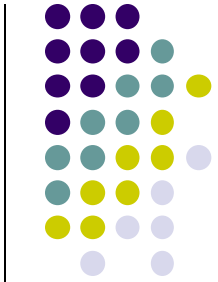
1. Sequence layer
2. Group of Pictures Layer
3. Picture Layer
4. Slice Layer
5. Macroblock Layer
6. Block Layer



[I B B P B B P B B] [I B B P B B P B B] [I B B P B B P B B] ….

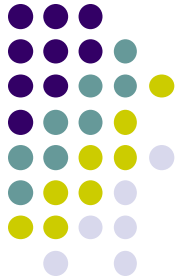
Picture (Frame) --> Slices --> MB's --> Blocks

MPEG-2



- **MPEG-2** is formally referred to as ISO/IEC specification 13818 and was completed in 1994
- MPEG-2 was developed to provide video quality not lower than and up to HDTV quality
  - Its target bit rates are 2 to 15 Mbps and it is optimized at 4 Mbps
  - The basic coding structure of MPEG-2 video is the same as that of MPEG-1
- A superset of MPEG-1 and backward compatible to the latter
  - With additional features to support interlaced video, panning window within a frame; and encoding at multiple quality levels, etc

MPEG-2 - 2

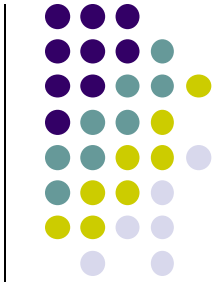


- Provides for four levels of video resolutions – low, main, high 1440, high

LEVEL	Pel/Line	Pels/Frame	Frame Rate (f/s)	Bit-rate (Mb/s)
High	1920	1152	60	80
High 1440	1440	1152	60	60
Main	720	576	30	15
Low	352	288	30	4

- Provides for 5 profiles – simple, main, spatial resolution, quantization accuracy, high
  - A Profile defines a new set of algorithms added as a superset to the algorithms in the profile that follows
- All profiles and levels are downward compatible:
  - Permits partial implementation defined by Profiles and Levels
  - This provides for scalability in the standard

Summary



- Discuss principles of inter-frame compression using block-based motion compensation
- Cover details of MPEG-1 and MPEG-2
- The motion vector may be exploited to extract approximate motion information of the video

Next Topic: H.261 and other video standards