IMAGE COMPRESSION (Module- IV)

By:-
Prof. Kamakshya Prasad Nayak
Department of CSE,
GCEK, Bhawanipatna

Motivation towards Compression

- Digital images require huge amounts of space for storage and large bandwidths for transmission.
  - A 640 x 480 color image requires close to 1MB of space.
- Calculate the space required for a SD (Standard Definition 720 * 480) movie of 2 hours running at 30 fps…!! (Answer = 224 GB)
- Imagine how movies came in two CD's / DVD's of very less size than required 224 GB size…?

Motivation Continued…

- Have you checked the size of any image when its clicked by your camera and compared it to the ones you share on social media…?
- Imagine the network congestion if all images shared over Whatsapp or any social media did not have any provision for compression.
- What would have been the cost of data transmission then?

Data Compression

- It aims at reducing the amount of data required to represent a given quantity of information.
- Data is the means by which information is conveyed.
- Information are said to contain **redundant** data.

Data vs Information

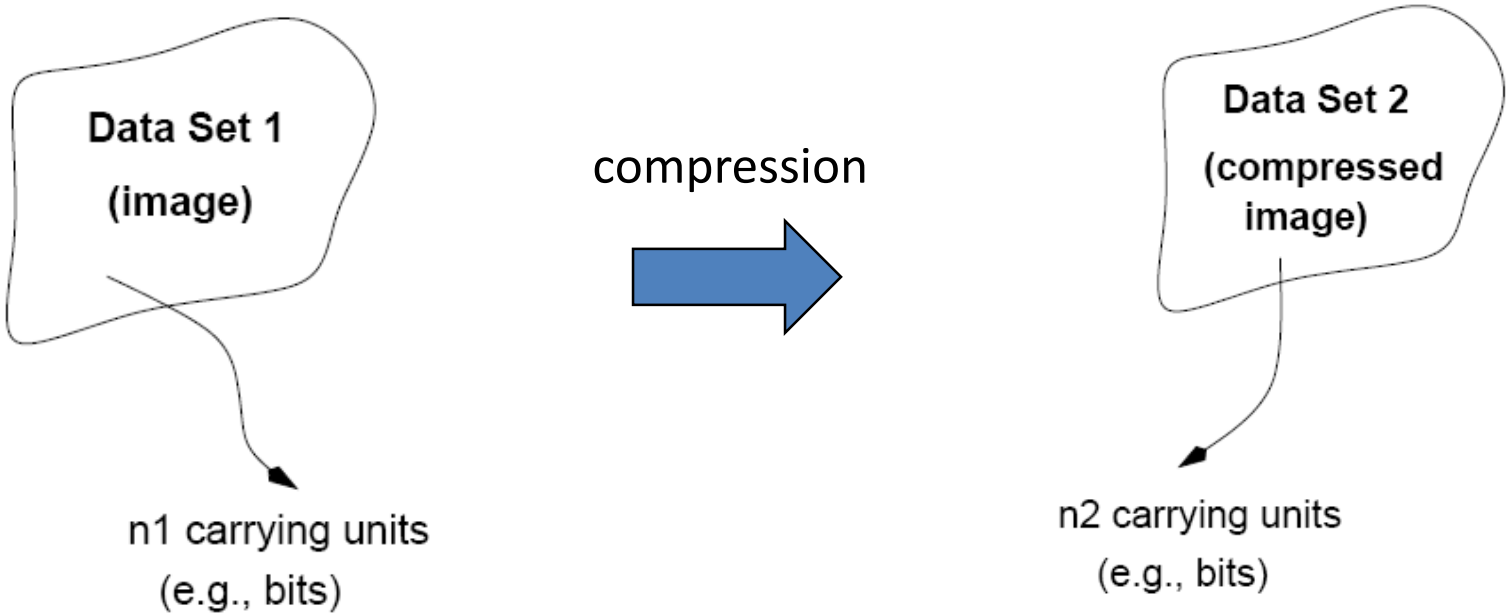
- The same amount of <u>information</u> can be represented by various amount of <u>data.</u>

- <u>For example:-</u>
- *Your wife, Helen, will meet you at Logan Airport in Boston at 5 minutes past 6:00 pm tomorrow night*

- *Your wife will meet you at Logan Airport at 5 minutes past 6:00 pm tomorrow night*

- *Helen will meet you at Logan at 6:00 pm tomorrow night*
- ***All 3 statements represent the same information with different levels of data redundancy, the first line containing maximum redundant data.***

Compression Ratio



Compression Ratio: $C_R = \frac{n_1}{n_2}$

Data Redundancy

- Relative data redundancy: $R_D = 1 - \frac{1}{C_R}$

Example:

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then C_R=1, R_D=0

if $n_2 \ll n_1$, then $C_R \to \infty$, $R_D \to 1$

if $n_2 \gg n_1$, then $C_R \to 0$, $R_D \to -\infty$

Types of Data Redundancy

(1) Coding

(2) Interpixel

(3) Psychovisual

- Compression attempts to reduce one or more of these redundancy types.

Image Compression

- It is the art and science of reducing the amount of data required to represent an image.

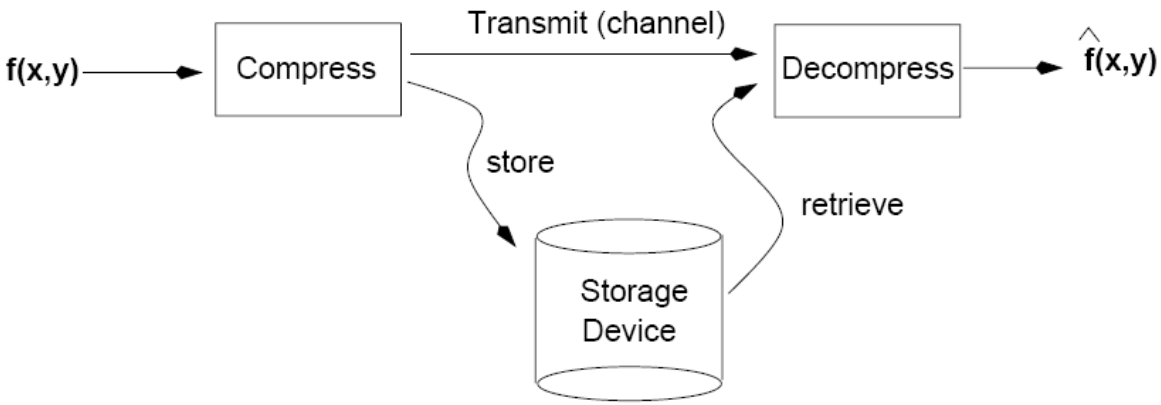


Fig 1: Idea of Compression

Coding Redundancy

- A **code** is a system of symbols used to represent a body of information or set of events.
- Each piece of information or event is assigned a sequence of code symbols called **code word**.
- The number of symbols in each codeword is called its **length**.
- For eg:- Each pixel in a gray scale image is represented by 8-bits binary. (Pixel value = 255, code = 11111111, code length per pixel= 8)

Spatial & Temporal Redundancy

- Spatial Redundancy means the neighbouring pixels are correlated in a 2-D image and hence it is redundant to store all the pixel values.

- Temporal Redundancy is observed in case of videos where the adjacent frames are similar to each other i.e. the frames are correlated in time.

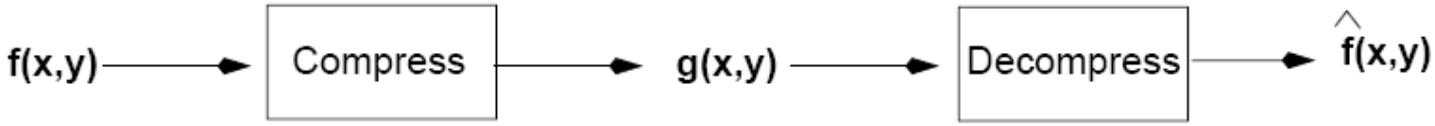
Psychovisual Redundancy

- The human eye does not respond with equal sensitivity to all visual information.
- It is more sensitive to the lower frequencies than to the higher frequencies in the visual spectrum.
- Idea: Discard data that is perceptually insignificant to human eyes.

Types of Image Compression

- Based on the loss incurred in the compression scheme, it is classified into the following:-
- Lossless
  - Information preserving
  - Low compression ratios
- Lossy
  - Not information preserving
  - High compression ratios
- <u>Trade-off:</u> image quality **vs** compression ratio

Fidelity Criteria



$$\hat{f}(x,y) = f(x,y) + e(x,y)$$

- How close is $f(x,y)$ to $\hat{f}(x,y)$?

- Criteria
  - <u>Subjective:</u> based on human observers
  - <u>Objective:</u> mathematically defined criteria

Subjective Fidelity Criteria

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Objective Fidelity Criteria

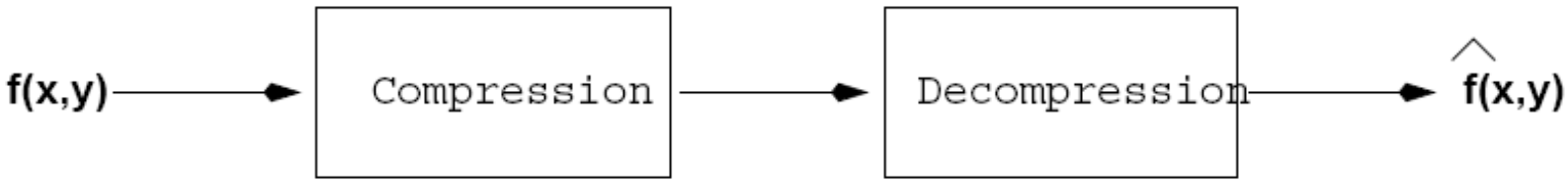
- Root mean square error (RMS)

$$e_{rms} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

- Mean-square signal-to-noise ratio (SNR)

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y))^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

Lossless Compression



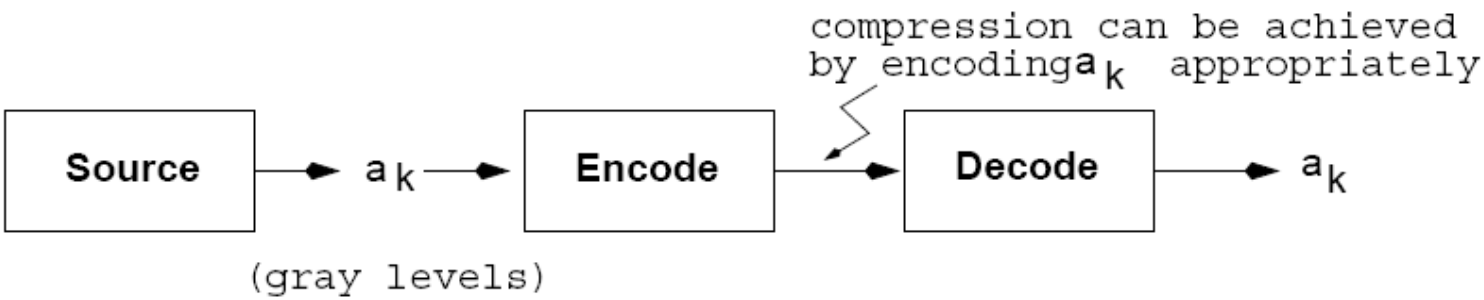
$$e(x, y) = \hat{f}(x, y) - f(x, y) = 0$$

Lossless Methods: Taxonomy

Types of Lossless Coding Techniques (Entropy Coding)

1. Repetitive Sequence Encoding (Ex:- Run Length Encoding)
2. Statistical Encoding (Ex:- Huffman, Arithmetic, LZW)
3. Lossless Predictive Coding (Ex:- DPCM)
4. Bit plane Encoding

Huffman Coding (coding redundancy)



- A variable-length coding technique.
- Optimal code (i.e., minimizes the number of code symbols per source symbol).
- <u>Assumption:</u> symbols are encoded one at a time!

Huffman Coding (cont'd)

- *Forward Pass*
  1. Sort probabilities per symbol
  2. Combine the lowest two probabilities
  3. Repeat *Step2* until only two probabilities remain.

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Huffman Coding (cont'd)

Backward Pass

Assign code symbols going backwards

Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Huffman Coding (cont'd)

- L_{avg} (Bits per symbol) using Huffman coding:

$$L_{avg} = E(l(a_k)) = \sum_{k=1}^{6} l(a_k)P(a_k) =$$

$$3x0.1 + 1x0.4 + 5x0.06 + 4x0.1 + 5x0.04 + 2x0.3 = 2.2 \text{ bits/symbol}$$

- L_{avg} assuming binary codes:

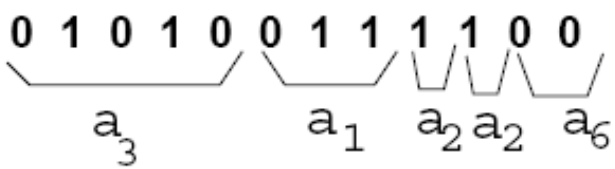
6 symbols, we need a 3-bit code

$$(a_1\text{: } 000,\ a_2\text{: } 001,\ a_3\text{: } 010,\ a_4\text{: } 011,\ a_5\text{: } 100,\ a_6\text{: } 101)$$

$$L_{avg} = \sum_{k=1}^{6} l(a_k)P(a_k) = \sum_{k=1}^{6} 3P(a_k) = 3\sum_{k=1}^{6} P(a_k) = 3 \text{ bits/symbol}$$

Huffman Coding/Decoding

- After the code has been created, *coding/decoding* can be implemented using a look-up table.
- Note that decoding is done unambiguously in this case.



Original source		
Sym.	Prob.	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

Arithmetic (or Range) Coding
(coding redundancy)

- No assumption on encoding source symbols one at a time.
  - Sequences of source symbols are encoded together.
  - There is no one-to-one correspondence between source symbols and code words.

- Slower than Huffman coding but typically achieves better compression.

Arithmetic Coding (cont'd)

- A sequence of source symbols is assigned a single arithmetic code word which corresponds to a sub-interval in [0,1].

- As the number of symbols in the message increases, the interval used to represent it becomes smaller.

- Smaller intervals require more information units (i.e., bits) to be represented.

Arithmetic Coding (cont'd)

Source Symbol	Probability
a_1	0.2
a_2	0.2
a_3	0.4
a_4	0.2

Encode message: $a_1\ a_2\ a_3\ a_3\ a_4$

1) Assume message occupies [0, 1)

0 1

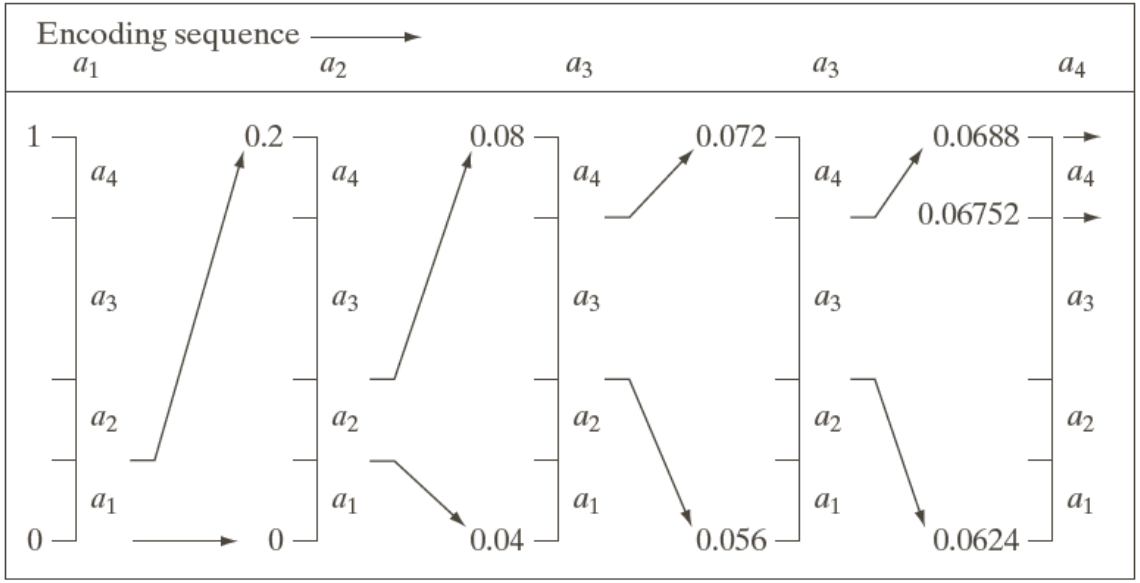
2) Subdivide [0, 1) based on the probability of α_i

Initial Subinterval
[0.0, 0.2)
[0.2, 0.4)
[0.4, 0.8)
[0.8, 1.0)

3) Update interval by processing source symbols

Example of Arithmetic Coding

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Encode

$a_1\ a_2\ a_3\ a_3\ a_4$

[0.06752, 0.0688)

or,

0.068

Example contd…

- The message $a_1\ a_2\ a_3\ a_3\ a_4$ is encoded using 3 decimal digits or 3/5 = 0.6 decimal digits per source symbol.

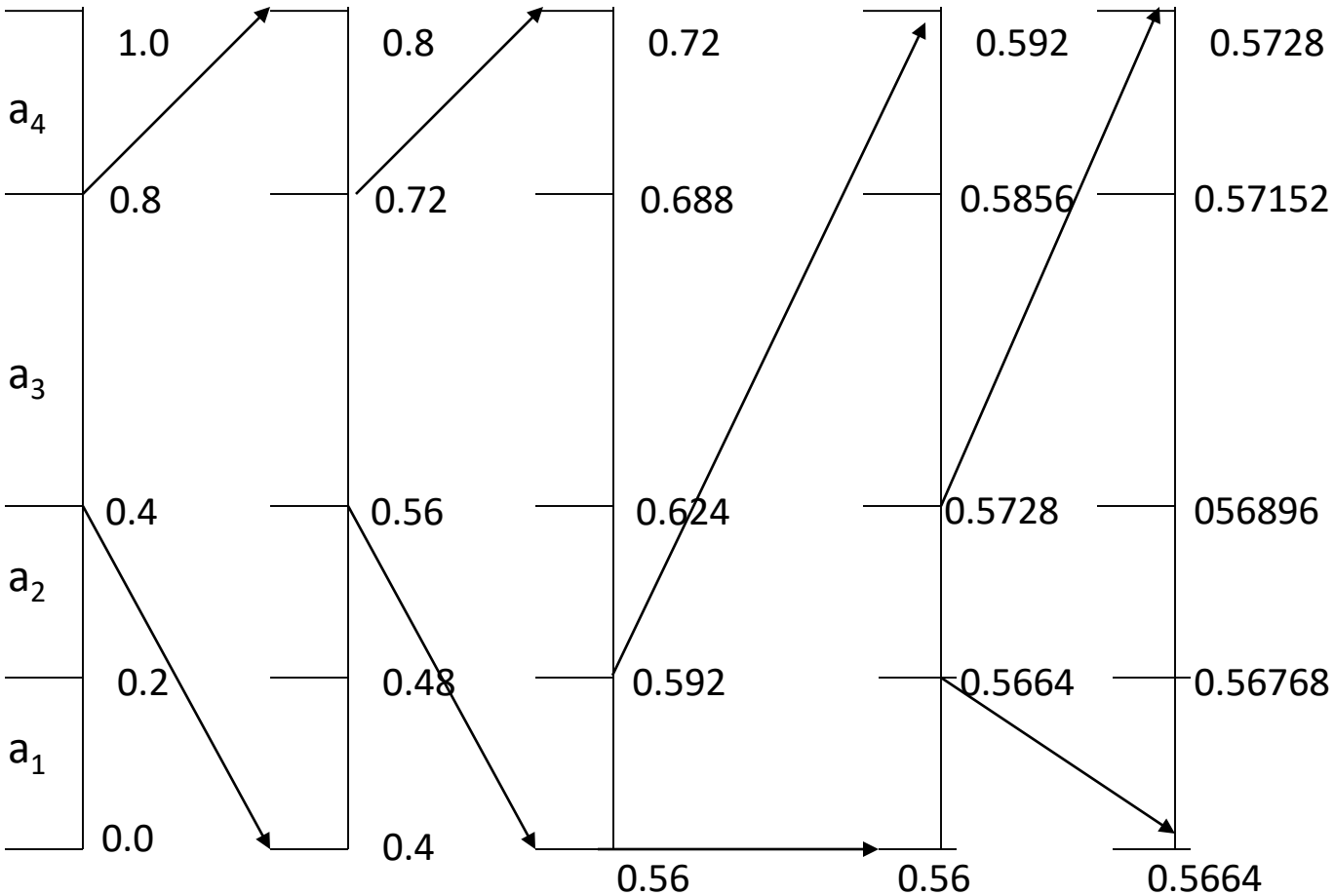
- The entropy of this message is:

$$H = -\sum_{k=0}^{3} P(r_k) log(P(r_k))$$

$-(3 \times 0.2\log_{10}(0.2)+0.4\log_{10}(0.4))=0.5786$ digits/symbol

<u>Note:</u> finite precision arithmetic might cause problems due to truncations!

Arithmetic Decoding



Decode 0.572

$a_3\ a_3\ a_1\ a_2\ a_4$

LZW Coding (Interpixel Redundancy)

- Requires no prior knowledge of pixel probability distribution values.
- Assigns fixed length code words to variable length sequences.
- It is a type of **Dictionary based Coding.**
- Included in GIF and TIFF and PDF file formats

LZW Coding

- A codebook (or dictionary) needs to be constructed.

- Initially, the first 256 entries of the dictionary are assigned to the gray levels 0,1,2,..,255 (i.e., assuming 8 bits/pixel)

Consider a 4x4, 8 bit image

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Initial Dictionary

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

LZW Coding (cont'd)

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

As the encoder examines image pixels, gray level sequences (i.e., **blocks**) that are not in the dictionary are assigned to a new entry.

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	39-39
511	-

- Is 39 in the dictionary……..Yes
- What about 39-39……No
- Then add 39-39 in entry 256

Example

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

CR = empty

If CS is found:

(1) No Output

(2) CR=CS

else:

(1) Output D(CR)

(2) Add CS to D

(3) CR=P

Concatenated Sequence: CS = CR + P

Currently Recognized Sequence (CR)	Pixel Being Processed (P)	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Decoding LZW

- The dictionary which was used for encoding need not be sent with the image.

- Can be built on the “fly” by the decoder as it reads the received code words.

Run-length coding (RLC)
(Interpixel redundancy)

- Used to reduce the size of a repeating string of characters (i.e., runs):

1 1 1 1 1 0 0 0 0 0 0 1 → (1,5) (0, 6) (1, 1)

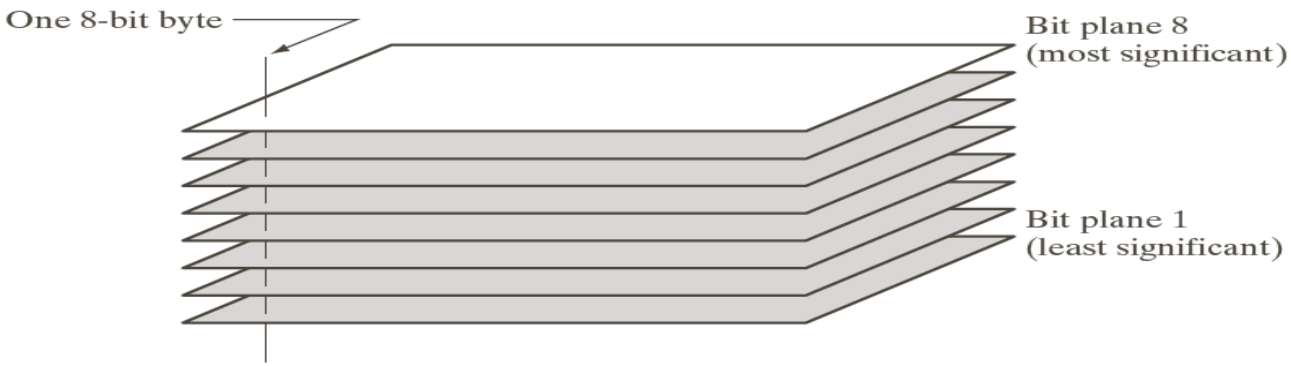
a a a b b b b b b c c → (a,3) (b, 6) (c, 2)

- Encodes a run of symbols into two bytes: **(symbol, count)**
- Can compress any type of data but cannot achieve high compression ratios compared to other compression methods.

Bit-plane coding (Interpixel redundancy)

- An effective technique to reduce inter pixel redundancy is to process each bit plane individually.

(1) Decompose an image into a series of binary images.



(2) Compress each binary image (e.g., using run-length coding)

Combining Huffman Coding with Run-length Coding

- Assuming that a message has been encoded using Huffman coding, additional compression can be achieved using run-length coding.

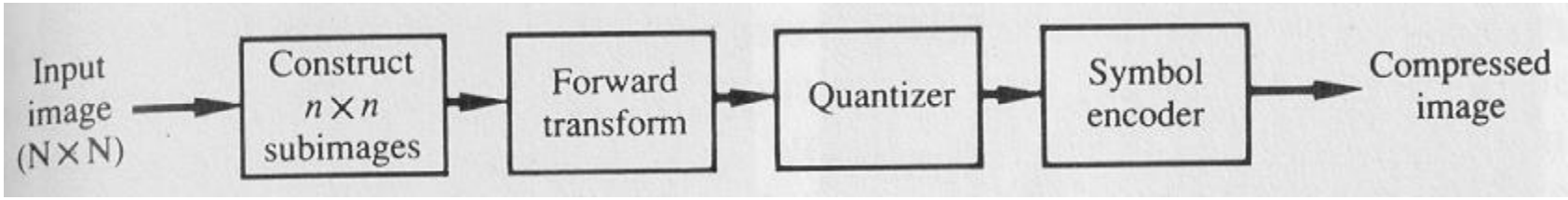
0 1 0 1 0 0 1 1 1 1 0 0

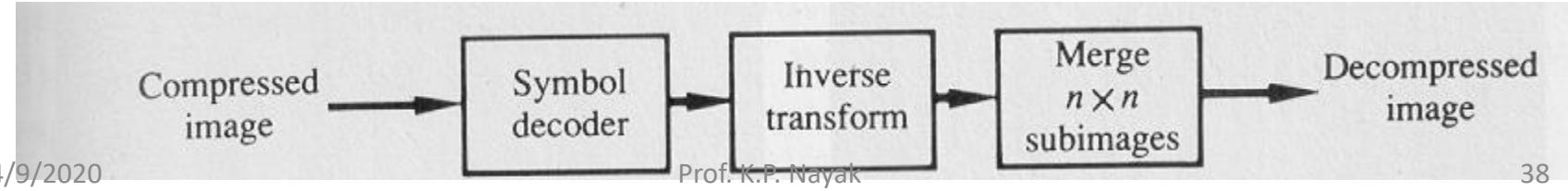
e.g., (0,1)(1,1)(0,1)(1,0)(0,2)(1,4)(0,2)

Transform based Coding

- Transform the image into a domain where compression can be performed more efficiently (i.e. reduce interpixel redundancies).
- Transform types:- DFT, DCT, K-L Transform

~ $(N/n)^2$ subimages





Transform selection

-Consider an image f (x,y) of size N X N

-The forward discrete transform is defined as

$$T(u,v) = \sum_{X=0}^{N-1}\sum_{y=O}^{N-1} f(x,y)g(x,y,u,v)$$

$$u,v = 0,1,\ldots\ldots, N-1$$

- **The inverse discrete transform is defined as**

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)h(x,y,u,v)$$

$$x,y = 0,1,\ldots\ldots\ldots\ldots, N-1$$

Example : Discrete Fourier Transform

- Forward transform kernel

$$g(x,y,u,v)=e^{j\frac{2\Pi}{N}(ux+vy)}$$

- Inverse transform kernel

$$h(x,y,u,v)=\frac{1}{N^2}e^{j\frac{2\Pi}{N}(ux+vy)}$$

Discrete cosine transform(DCT)

The discrete cosine transform (DCT) gets its name from the fact that the rows of the N x N transform matrix C are obtained as a function of cosines.

$$g(x,y,u,v) = h(x,y,u,v)$$

$$= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\prod}{2N}\right]\cos\left[\frac{(2y+1)v\prod}{2N}\right]$$

Where

$$\alpha(u) = \sqrt{\frac{1}{N}} \text{ for } u = 0$$

$$= \sqrt{\frac{2}{N}} \text{ for } u = 1, 2 \ldots\ldots\ldots, N-1$$

Why DCT

-Blocking artifacts are less pronounced in the DCT than the DFT.

-The DCT is a good approximation of the Karhunen-Loeve Transform(KLT) which is optimal in terms of energy compaction.

-However unlike KLT , the DCT has image independent basis functions.

-The DCT is used in JPEG compression standard.

JPEG (Joint Photographic Experts Group)

-> It is a compression standard for still images

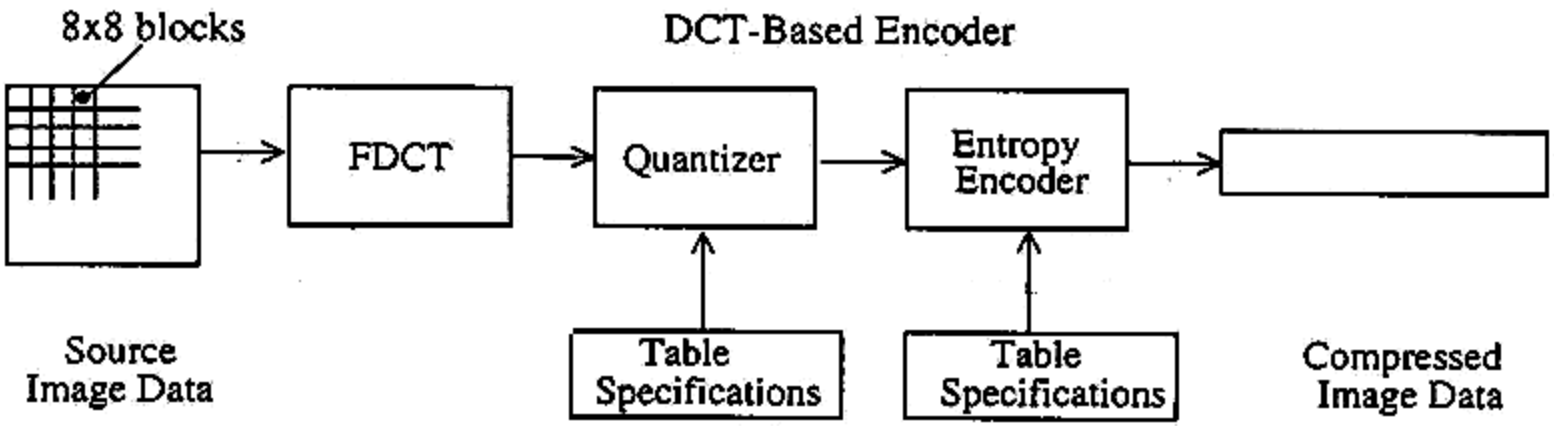
-> It defines three different coding systems :

1. A lossy baseline coding system based on DCT (adequate for most compression applications)

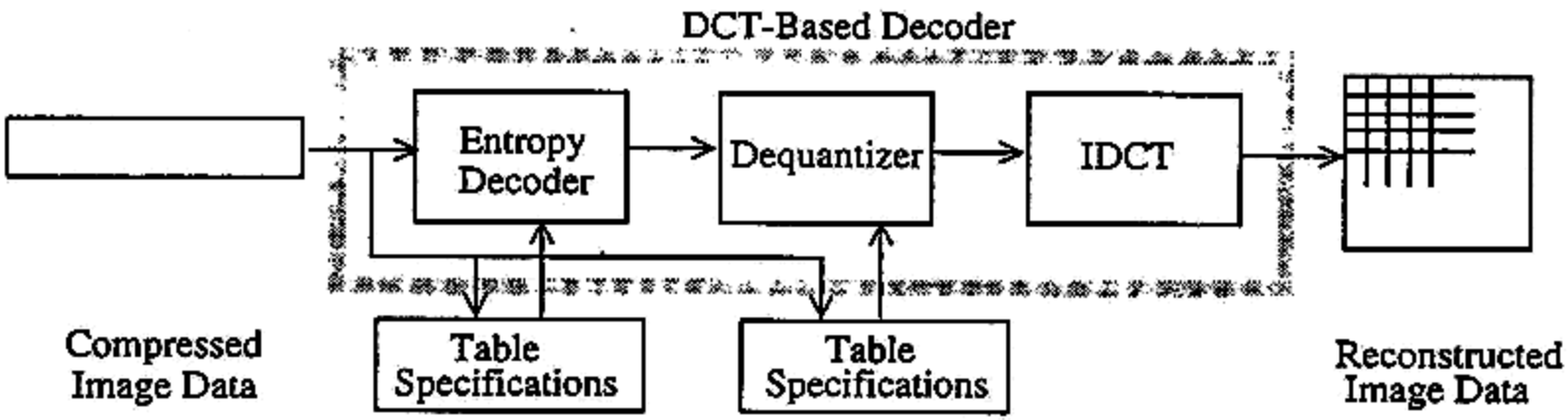
2. An extended system for greater compression, higher precision, or progressive reconstruction applications

3. A lossless independent coding system for reversible compression.

Steps of JPEG Compression



DCT-Based Encoder Processing Steps



JPEG – Baseline Coding System

1. Computing the DCT: the image is divided into 8 X 8 blocks The blocks of component are then transformed into frequency components.
2. Each block is represented by 64 frequency components, one of which (the DC component) is the average color of the 64 pixels in the block.
3. Applying FDCT to change the blocks from spatial domain into frequency domain.
4. Roughly speaking , F(0,0) is the average value within the block (called DC component).
5. Other values stores the variation within a block (AC components).

Example Fast DCT

f =

183	160	94	153	194	163	132	165
183	153	116	176	187	166	130	169
179	168	171	182	179	170	131	167
177	177	179	177	179	165	131	167
178	178	179	176	179	164	130	171
179	180	180	179	182	164	130	171
179	179	180	182	183	170	129	173
180	179	181	179	181	170	130	69

Example : Level Shifting

	55	32	-34	25	66	35	4	37
	55	25	-12	48	59	38	2	41
	51	40	43	54	51	42	3	39
fs=f-128	49	49	51	49	51	37	3	39
	50	50	51	48	54	36	2	43
	51	52	52	51	55	36	2	43
	51	51	52	54	55	42	1	45
	52	51	53	51	53	42	2	41

Example : Computing the DCT

dcts=round(dct2(fs))

312	56	-27	17	79	-60	26	-26
-38	-28	13	45	31	-1	-24	-10
-20	-18	10	33	21	-6	-16	-9
-11	-7	9	15	10	-11	-13	1
-6	1	6	5	-4	-7	-5	5
3	3	0	-2	-7	-4	1	2
3	5	0	-4	-8	-1	2	4
3	1	-1	-2	-3	-1	4	1

Example : The Normalization Matrix

qmat=

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

Example :

Normalization Table

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

312/16
= 20

312	56	-27	17	79	-60	26	-26
-38	-28	13	45	31	-1	-24	-10
-20	-18	10	33	21	-6	-16	-9
-11	-7	9	15	10	-11	-13	1
-6	1	6	5	-4	-7	-5	5
3	3	0	-2	-7	-4	1	2
3	5	0	-4	-8	-1	2	4
3	1	-1	-2	-3	-1	4	1

20	5	-3	1	3	-2	1	0
-3	-2	1	2	1	0	0	0
-1	-1	1	1	1	0	0	0
-1	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	0	0	0	0	0

Quantized value

Example : Thresholding

t=round(dcts. /qmat)=

20	5	-3	1	3	-2	1	0
-3	-2	1	2	1	0	0	0
-1	-1	1	1	1	0	0	0
-1	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	0	0	0	0	0

Zig- Zag Scanning of the Coefficients

t= round(dcts./qmat)=

20	5	-3	1	3	-2	1	0
-3	-2	1	2	1	0	0	0
-1	-1	1	1	1	0	0	0
-1	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	0	0	0	0	0

[20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,0,EOB]

Decoding the Coefficients

t X qmat=

320	55	-30	16	72	-80	51	0
-36	-24	14	38	26	0	0	0
-14	-13	16	24	40	0	0	0
-14	0	0	29	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Computing the IDCT

fs_hat=round(idct2(ds_hat))

67	12	-9	20	69	43	-8	42
58	25	15	30	65	40	-4	47
46	41	44	40	59	38	0	49
41	52	59	43	57	42	3	42
44	54	58	40	58	47	3	33
49	52	53	40	61	47	1	33
53	50	53	46	63	41	0	45
55	50	56	53	64	34	-1	57

Shifting Back the Coefficients

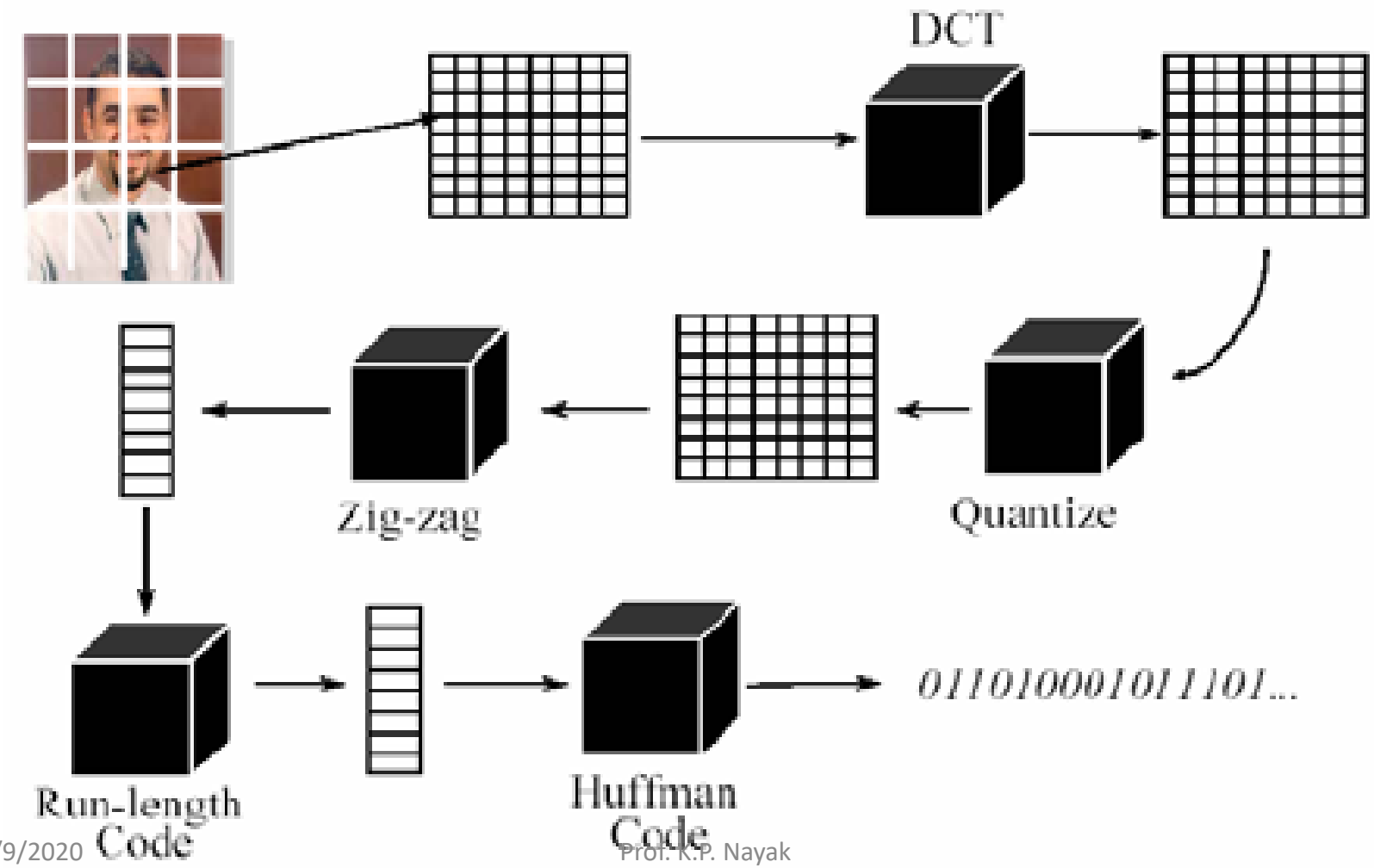
f_hat=fs_hat+128

195	140	119	148	197	171	120	170
186	153	143	158	193	168	124	175
174	169	172	168	187	166	128	177
169	180	187	171	185	170	131	170
172	182	186	168	186	175	131	161
177	180	181	168	189	175	129	161
181	178	181	174	191	169	128	173
183	178	184	181	192	162	127	185

f =

183	160	94	153	194	163	132	165
183	153	116	176	187	166	130	169
179	168	171	182	179	170	131	167
177	177	179	177	179	165	131	167
178	178	179	176	182	164	130	171
179	180	180	179	183	164	130	171
179	179	180	182	183	170	129	173
180	179	181	179	181	170	130	169

JPEG as Block Transform Encoding



What is Wavelet Transform ?

-Wavelets are functions defined over a finite interval and having an average value of zero.

- The basic idea of the wavelet transform is to represent any arbitrary function (t) as a superposition of a set of such wavelets or basis functions.

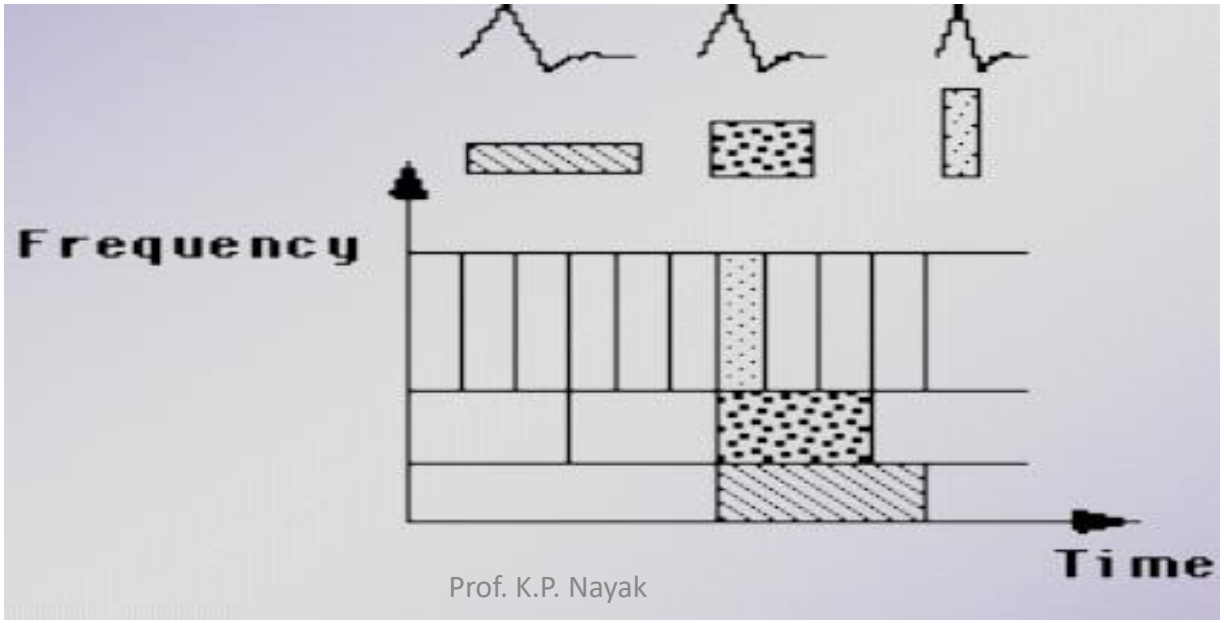
- These basis function or baby wavelets are obtained from a single prototype wavelet called mother wavelet , by contractions (scaling) and translations (shifts).

- The Discrete Wavelet Transform of a finite length signal x(n) having N components , for example , is expressed by an N x N matrix.

Why Wavelet-based Compression?

- No need to divide input image into blocks and its basis functions have variable length to avoid blocking artifacts.
- More robust under transmission and decoding errors.
- Better matched to the human visual system(HVS) characteristics

❖ Good frequency resolution at lower frequencies, good time resolution at higher frequencies – good for natural images.



Subband coding

-The fundamental concept behind Subband Coding (SBC) is to split up the frequency band of a signal (image in our case) and then to code each subband using a coder and bit rate accurately matched to the statistics of the band.

- SBC has been used extensively first in speech coding and later in image coding.

Wavelets as filter

Essentially a multiresolution filtering process:

– HPF (Mother wavelet)

– LPF (Father scaling)

– Apply in different resolutions

One Stage Filtering

Approximations and details:

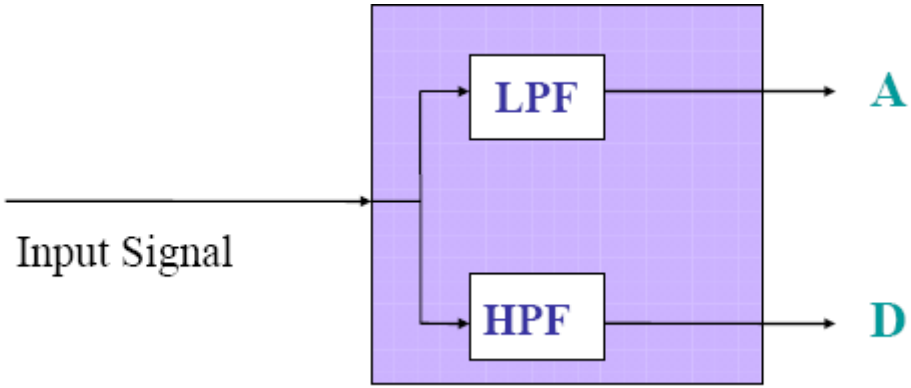
- The low-frequency content is the most important part in many applications, and gives the signal its identity.
This part is called ***"Approximations".***

- The high-frequency gives the 'flavor', and is called ***"Details".***

Approximations and Details:

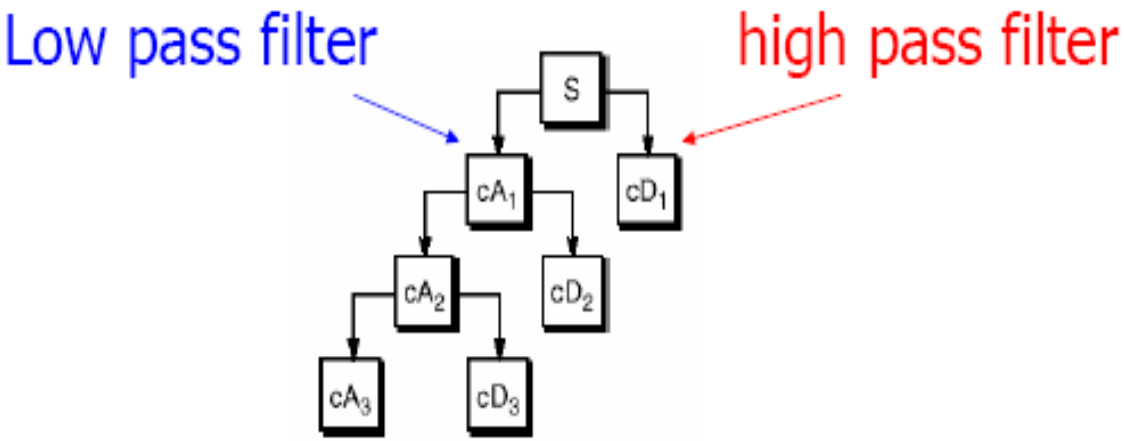
- **Approximations:** low- frequency components of the signal

- **Details:** high- frequency components

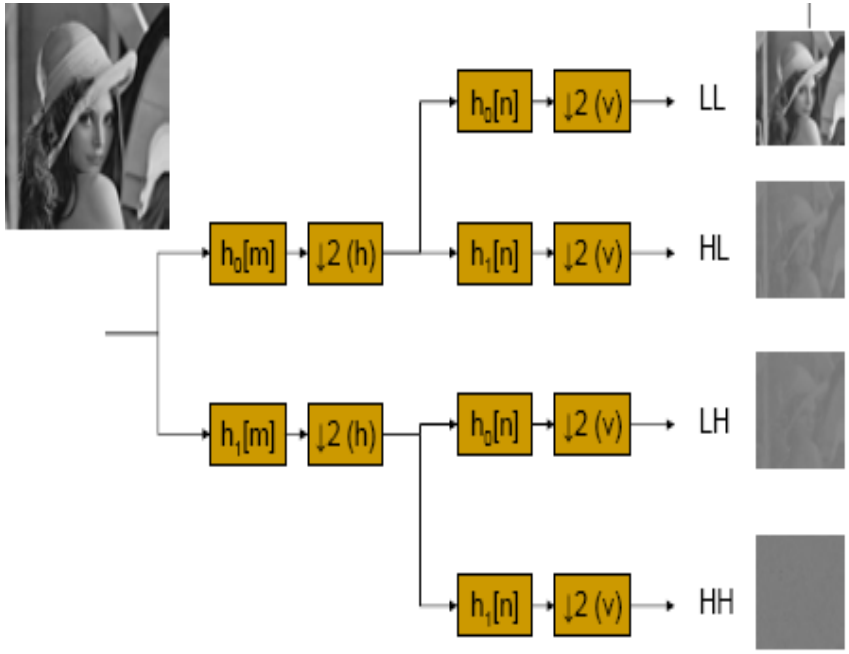


Multi-level Decomposition

Iterating the decomposition process, breaks the input signal into many lower-resolution components.
Wavelet decomposition tree *is given below:-*

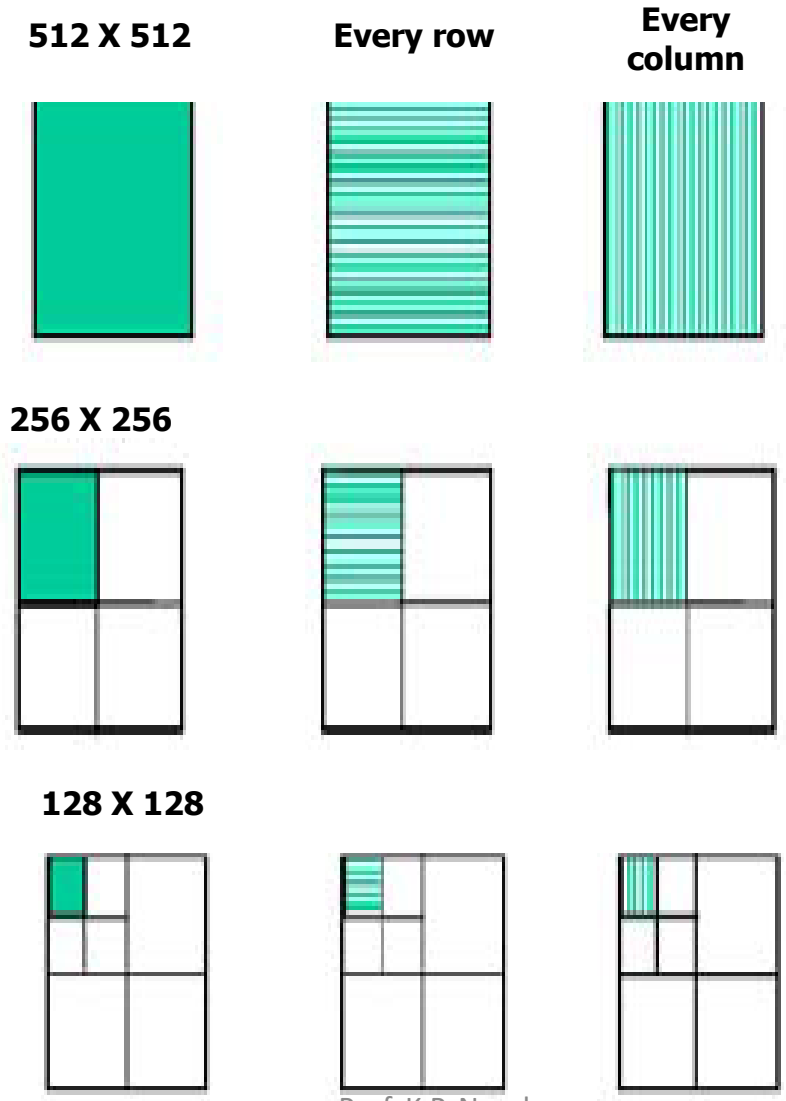


Wavelet Image Decomposition

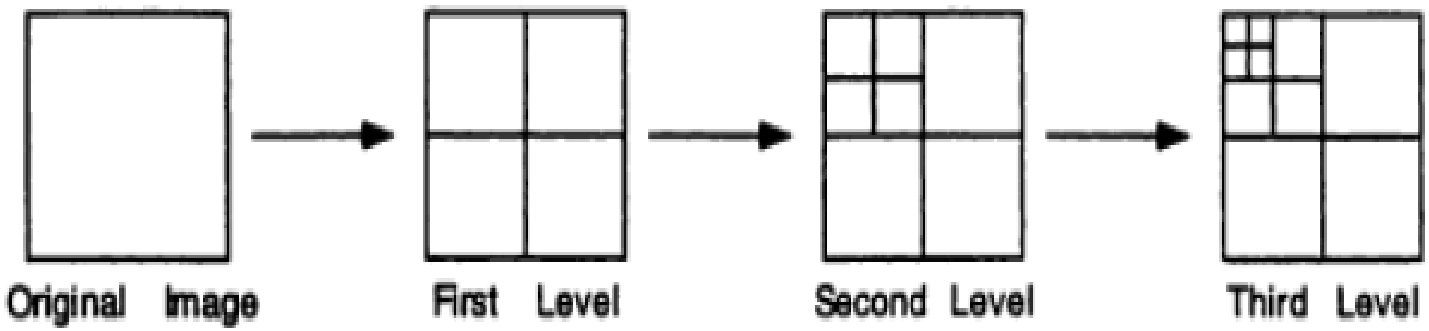


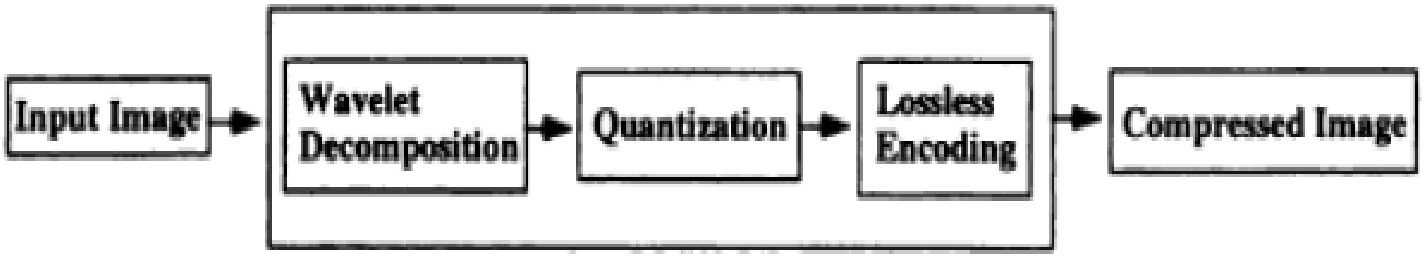
Wavelets Image decomposition

- **Multilevel Decomposition**



Decomposition and Compression





Quantization

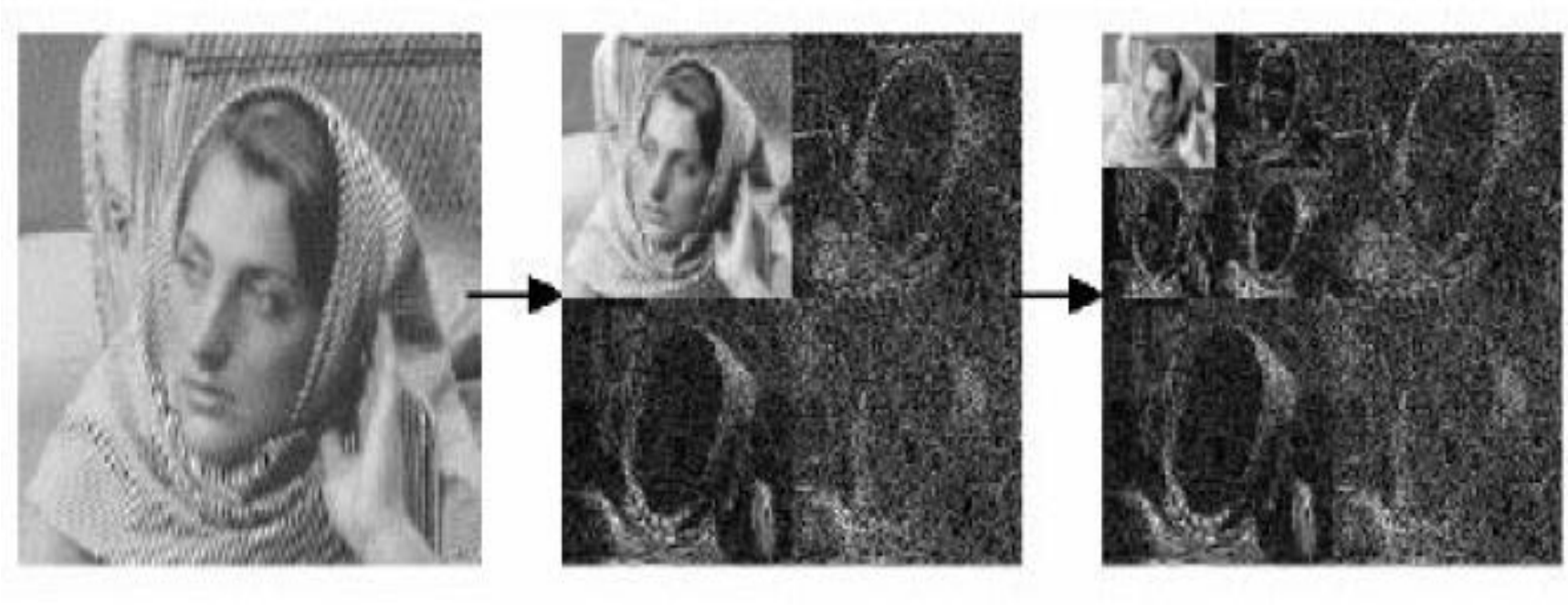
Two important observations:

1. Natural images in general have a low pass spectrum, so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands.

2. Large wavelet coefficients are more important than smaller wavelet coefficients.

Example

- **Here is a two-stage wavelet decomposition of an image. Notice the large number of zeros (black):**



Matlab toolbox

Comparison

JPEG (DCT based)

JPEG-2000 (Wavelet based)

Comparison

JPEG-2000(1.83KB)

Original(979 KB)

JPEG (6.21 KB)

References

- Gonzalez R.C. and Woods R.E, “Digital Image Processing”, Pearson
- https://www.youtube.com/playlist?list=PLuv3GM6-gsE08DuaC6pFUvFaDZ7EnWGX8 (NPTEL video tutorial Part-1 by Dr. P.K. Biswas, IIT Kharagpur)
- https://www.youtube.com/playlist?list=PL1F076D1A98071E24 (NPTEL video tutorial Part-2 by Dr. P.K. Biswas, IIT Kharagpur)