Digital Image Processing 15CS753

MODULE-5

Image Compression

- Module – 5 Image Compression: 8 Hours

- Introduction, coding Redundancy ,
- Inter-pixel redundancy, image compression model,
- Lossy and Lossless compression,
- Huffman Coding, Arithmetic Coding,
- LZW coding, Transform Coding,
- Sub-image size selection, blocking,
- DCT implementation using FFT,
- Run length coding.

Image Compression

- Image compression is art and science of reducing amount of data required to represent the image.
- Need for image compression ?
- To reduce the storage space
- To reduce the transmission time while web surfing
- Fundamentals:
- Data compression refers to the process of reducing the amount of data required to represent the given information
- Data and information are not same
- Data is a mean to convey the information
- W.k.t. same information can be represented using various amount of data
- Some representations may contain irrelevant or repeated information – redundant data

Image Compression

- Let there be two representations to of the same information and let a and b be the number of bits to represent them respectively.
- The relative data redundancy R_D is given by,

$$R_D = 1 - \frac{1}{C_R}$$

- where C_R is compression ratio, given by $C_R = a/b$
- Based on a and b values there can be three different cases
- i. If b = a, no redundant data is present in first set
- ii. If b << a, highly redundant data is present in first set.
- iii. If b >> a, highly redundant data is present in second set.

Image Compression

- In image compression context, a in the equation for C_R, usually is the number of bits needed to represent an image as 2-D array of intensities
- Typically 2-D array of intensities suffer from three types of redundancies
  - Coding Redundancy
  - Spatial or temporal redundancy
  - Irrelevant information
- Coding Redundancy:
- Code- a system of symbols to represent body of information
- Each piece of info is assigned with sequence of code symbols called as code words
- Number of symbols in each code word is – length
- The 8-bit codes that are used to represent the intensities in most of the 2-D intensity arrays contain more bits than needed to represent the intensities

Image Compression

- Spatial or temporal redundancy
- Most of the pixels in 2-D intensity arrays are correlated spatially (each pixel is similar to dependent on neighboring pixels) info is unnecessarily replicated in the representation of correlated pixels
- Irrelevant information
- Most of the 2-D intensity arrays contain information that is ignored by the human visual system or is extraneous to the use of the image.
- Hence they can be called as redundant as they are not used

- Image compression means eliminating one or more redundancy in the image

Image Compression

- Coding redundancy
- Let us assume that, a discrete random variable r_k, in the interval [0, L-1] is used to represent the intensities of an MXN image and that each rk, occurs with probability of $p_r(r_k)$
- Then we can write $p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, \ldots, L-1$ ….(1)
- Where L is number of intensity values and n_k is number of occurrences of kth intensity
- If no. of bits used to represent each value of r_k is $l(r_k)$, then average no. of bits needed to represent each pixel is given by
- $$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$ …….(2)
-
- Average length of code word assigned to various intensity values is given by the sum of product of no. of bits used represent each intensity and their probability of occurrences
- Total no. of bits needed to represent MXN image is MNL_{avg}.

Image Compression

- If the intensities are represented using m-bit fixed length code the RHS of the previous equation reduces to m, while substituting $l(r_k) = m$.
- By taking constant m out of summation we are left with only summation of $p_r(r_k)$ which is always 1
- Consider the following computer generated 256 X 256 image

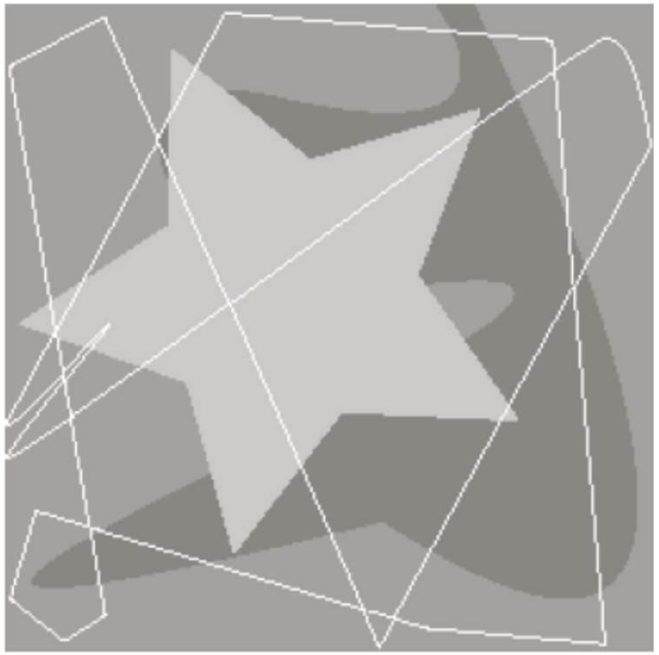
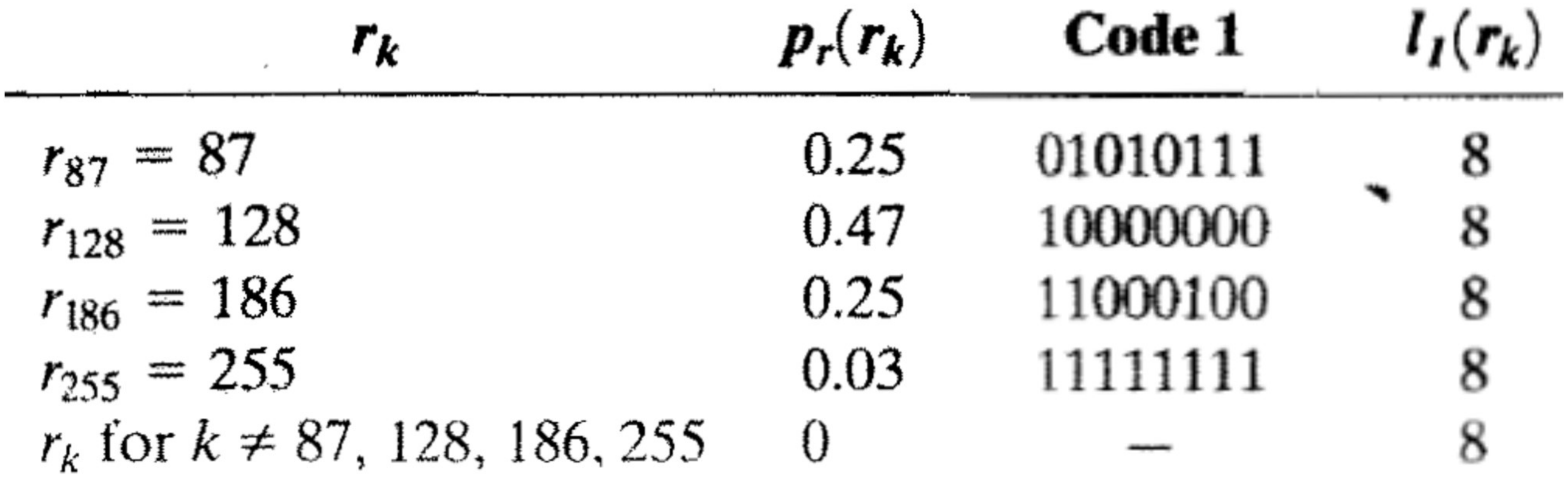


Image Compression

- The intensity distribution for the same is as given below



r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$
$r_{87} = 87$	0.25	01010111	8
$r_{128} = 128$	0.47	10000000	8
$r_{186} = 186$	0.25	11000100	8
$r_{255} = 255$	0.03	11111111	8
r_k for $k \neq 87, 128, 186, 255$	0	—	8

- We can see that, there are 4 different intensities
- If we choose natural 8-bit binary code (say code-1)to represent these intensities, L_{avg}, the average number of bits for code-1will be 8 bits

Image Compression

- Now suppose that we choose another scheme say code 2 as shown below

r_k	$p_r(r_k)$	Code 2
$r_{87} = 87$	0.25	01
$r_{128} = 128$	0.47	1
$r_{186} = 186$	0.25	000
$r_{255} = 255$	0.03	001
r_k for $k \neq 87, 128, 186, 255$	0	—

- Let us compute the average code length based on the equation 2

$$L_{avg} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

Image Compression

- Total number of bits needed to represent the entire image is MNL_{avg} = 256 X 256 X 1.81
- = 1,18,621 bits
- Using the equations of compression ratio and redundancy we can write
-
- $C = \frac{256 \times 256 \times 8}{118,621} = \frac{8}{1.81} \approx 4.42$ And $R = 1 - \frac{1}{4.42} = 0.774$
- This means that, 77% of the data in the original 8-bit image as redundant.
- The compression in code-2 is achieved by assigning lesser bits to the more probable intensity values than the less probable ones.
- The resulting code will be variable length code

r_k	$p_r(r_k)$	Code 2
$r_{87} = 87$	0.25	01
$r_{128} = 128$	0.47	1
$r_{186} = 186$	0.25	000
$r_{255} = 255$	0.03	001
r_k for $k \neq 87, 128, 186, 255$	0	–

Image Compression

- The intensity distribution for the same is as given below

r_k	$p_r(r_k)$	Code 1
$r_{87} = 87$	0.25	01010111
$r_{128} = 128$	0.47	10000000
$r_{186} = 186$	0.25	11000100
$r_{255} = 255$	0.03	11111111
r_k for $k \neq 87, 128, 186, 255$	0	—

- We can see that, there are 4 different intensities
- If we choose natural 8-bit binary code (say code-1)to represent these intensities, L_{avg}, the average number of bits for code-1will be 8 bits

Image Compression

- Now suppose that we choose another scheme say code 2 as shown below

r_k	$p_r(r_k)$	Code 2
$r_{87} = 87$	0.25	01
$r_{128} = 128$	0.47	1
$r_{186} = 186$	0.25	000
$r_{255} = 255$	0.03	001
r_k for $k \neq 87, 128, 186, 255$	0	—

- Let us compute the average code length based on the equation 2

$$L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

Image Compression

- Total number of bits needed to represent the entire image is MNL_{avg} = 256 X 256 X 1.81
- = 1,18,621 bits
- Using the equations of compression ratio and redundancy we can write
-
- $C = \frac{256 \times 256 \times 8}{118{,}621} = \frac{8}{1.81} \approx 4.42$ And $R = 1 - \frac{1}{4.42} = 0.774$
- This means that, 77% of the data in the original 8-bit image as redundant.
- The compression in code-2 is achieved by assigning lesser bits to the more probable intensity values than the less probable ones.
- The resulting code will be variable length code

r_k	$p_r(r_k)$	Code 2
$r_{87} = 87$	0.25	01
$r_{128} = 128$	0.47	1
$r_{186} = 186$	0.25	000
$r_{255} = 255$	0.03	001
r_k for $k \neq 87, 128, 186, 255$	0	–

Image Compression

- Fidelity Criteria:
- This is used evaluate the information loss during data compression
- Need:
- W.k.t. There is a loss of real or quantitative visual information when redundancy is removed.
- Thus, there is a chance for losing the information of interest.
- Therefore, there is a need to quantify the nature and extent of the information loss using repeatable and reproducible criteria.
- **Types:**
  - Objective Fidelity Criteria
  - Subjective Fidelity Criteria

Image Compression

- Objective Fidelity Criteria:
- The objective fidelity criterion expresses the level of information loss as a function of two parameters. They are
  - The original or input image
  - The compressed and successively decompressed output image
- RMS Error:
- The root-mean-square (rms) error between the input and output image is a very good example for objective fidelity criteria.

Image Compression

- The error between two images f(x,y) and $f^{\wedge}$(x,y) is given by

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

- The total error between two MxN size images is

$$e_{total} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]$$

- The root-mean-square error e_{rms} between $f^{\wedge}$(x,y) and f(x,y) is obtained by

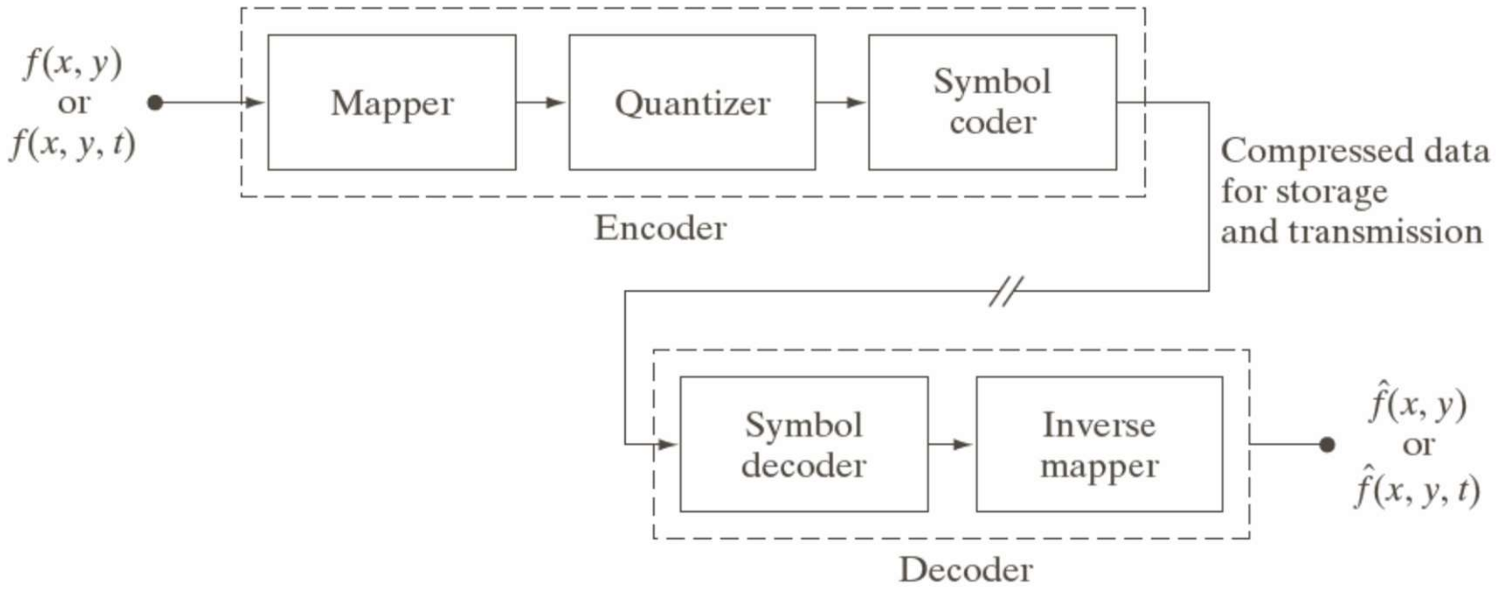
$$e_{rms} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]^2}$$

Image Compression

- Advantage:
- Simple and convenient technique to evaluate information loss
- Subjective Fidelity Criteria
- Since human perception is based on subjective quality, it is more suitable to use subjective fidelity criteria to evaluate the information loss.
- **Concept:**
- This method is implemented by showing a ‘typical’ decompressed image to a number of viewers.
- Then, their evaluation is averaged to take a decision.
- These evaluations may be done using a rating scale or side-by-side comparisons such as excellent, better, same, worse, etc.

Image Compression

- Image compression Model:
- Image compression system consists of two distinct functional units – encoder and decoder
- Encoder performs compression and decoder performs decompression.
- Schematically we can represent this system as shown below



$f(x, y)$ or $f(x, y, t)$
Mapper
Quantizer
Symbol coder
Encoder
Compressed data for storage and transmission
Symbol decoder
Inverse mapper
Decoder
$\hat{f}(x, y)$ or $\hat{f}(x, y, t)$

Image Compression

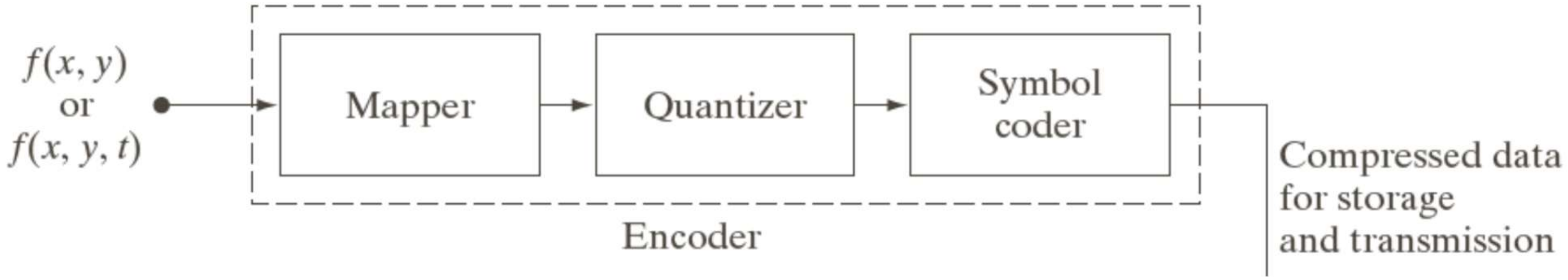
- Input image f(x, y) is fed to the encoder, which gives the compressed version of image.
- This will be stored for later use or transmitted for storage
- When the compressed image is given to decoder, a reconstructed output f^(x,y) will be generated
- If this is exact replica of f(x, y) then such a compression system is called lossless or error free system.
- If not, reconstructed image will be distorted and such a system is called lossy system
- Encoding or compression process:
- 

Image Compression

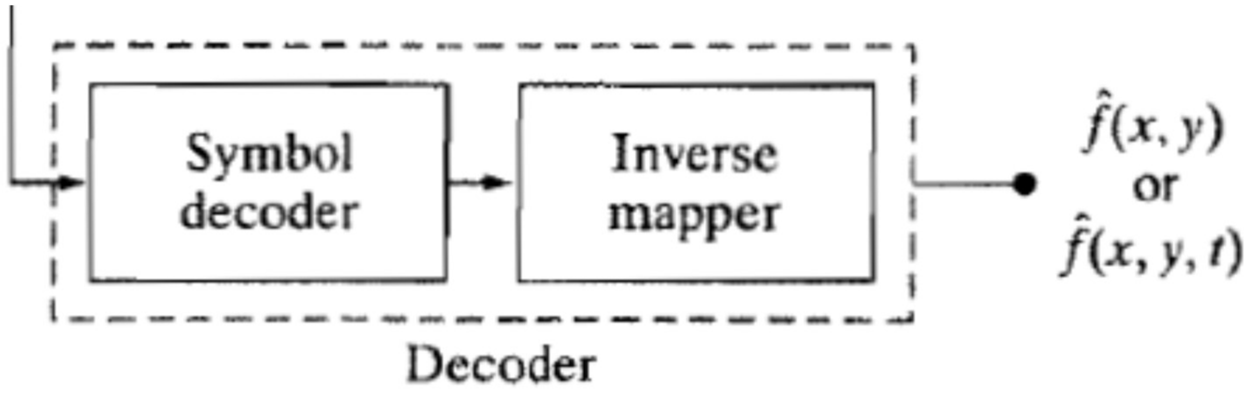
- The encoder is designed to remove the redundancies through three operations one by one
- The first stage is mapper stage, which transforms f(x, y) into a format designed to reduce the spatial and temporal redundancy
- This operation is reversible and may or may not reduce the amount of data required to represent the image
- Run-length coding is an example of mapping function
- The second stage is quantizer.
- This reduces the accuracy of the mapper’s output in accordance with the pre-established fidelity criterion
- Goal is to keep irrelevant information out of the compression
- Note that this is irreversible operation and hence to be skipped if error-free compression is needed

Image Compression

- Third stage is symbol coder.
- This generates fixed or variable length code to represent the quantizer output and maps the output in accordance with the code.
- In most of the cases, variable length code is used
- Shortest code words are assigned to most frequently occurring quantizer output values.
- This reduces the coding redundancy

Image Compression

- Decoding process:
- Decoder consists of two components – symbol decoder and inverse mapper



- They perform inverse operations of symbol coder and mapper
- Since quantization is irreversible process, reverse of this is not shown in decoding process

Image Compression

- Error Free Compression / Lossless Compression:
- Error-free compression is the acceptable data reduction method since there is no data loss.
- This method is applicable to both binary and gray-scale images.
- It provides compression ratios ranging from 2 to 10.
- Operations:
  - (i) Forming an alternative representation for the given image by which its interpixel redundancies are reduced
  - (ii) Coding the representation to remove its coding redundancies.
- E.g.:
- Variable-Length Coding
- Bit-Plane coding
- LZW Coding
- Lossless Predictive Coding

Image Compression

- Types of Compression
- Two broad categories
- 1. Lossless algorithms.
- 2. Lossy algorithms.
- Lossless data compression:
- The original data can be recovered exactly from the compressed data.
- Generally used for application where any difference between the compressed and the reconstructed data cannot be allowed.
- These techniques generally are composed of relatively two independent operations
- 1. A representation in which its inter-pixel redundancies are reduced.
- 2. Codification of the representation to eliminate coding redundancies.

Image Compression

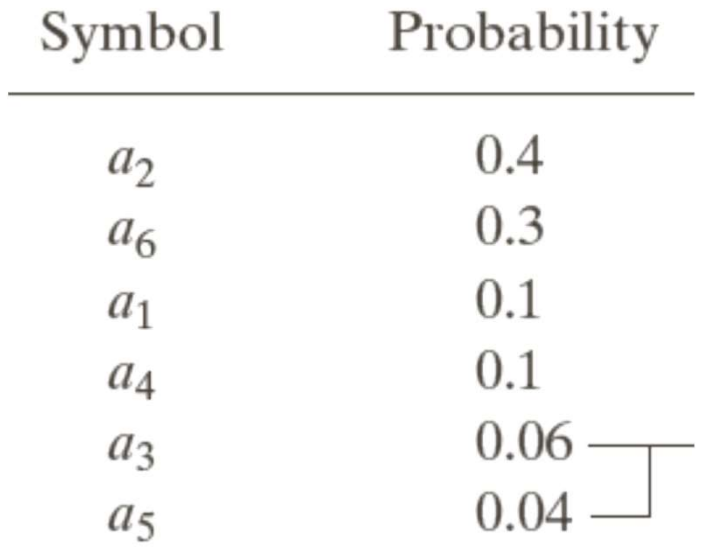
- E.g.: Variable length coding, Huffman coding, Arithmetic coding, LZW coding, Bit plane coding, Lossless Predictive coding
- They provide a compression ratio of 2 to 10 and they are equally applicable to both binary to gray scale images
- Lossy compression techniques:
- Involve some loss of information and data cannot be recovered from the same.
- These are used where the some loss of data can be acceptable.
- E.g.: Compressed Video signal is different from the original.
- However, we can generally obtain the higher compression ratio than the Lossless compression methods.
- Commonly used lossy compression techniques are Lossy predictive coding, Transform coding, Zonal coding, Wavelet coding, Image compression standard.
- These techniques are much more effective at compression than the Lossless methods.

Image Compression

- Huffman coding:
- Most popular coding scheme used for removal of coding redundancy
- Developed by D.A. Huffman in 1952 and these are optimal codes that map one symbol to one code word.
- Huffman coding is used to eliminate coding redundancy.
- It is a variable length coding and is used in lossless compression.
- Huffman encoding assigns smaller codes for more frequently used symbols and larger codes for less frequently used symbols.
- Variable-length coding table can be constructed based on the probability of occurrence of each source symbol.
- Its outcome is a prefix code that expresses most common source symbols with shorter sequence of bits. The more probable of occurrence of any symbol the shorter is its bit representation.

Image Compression

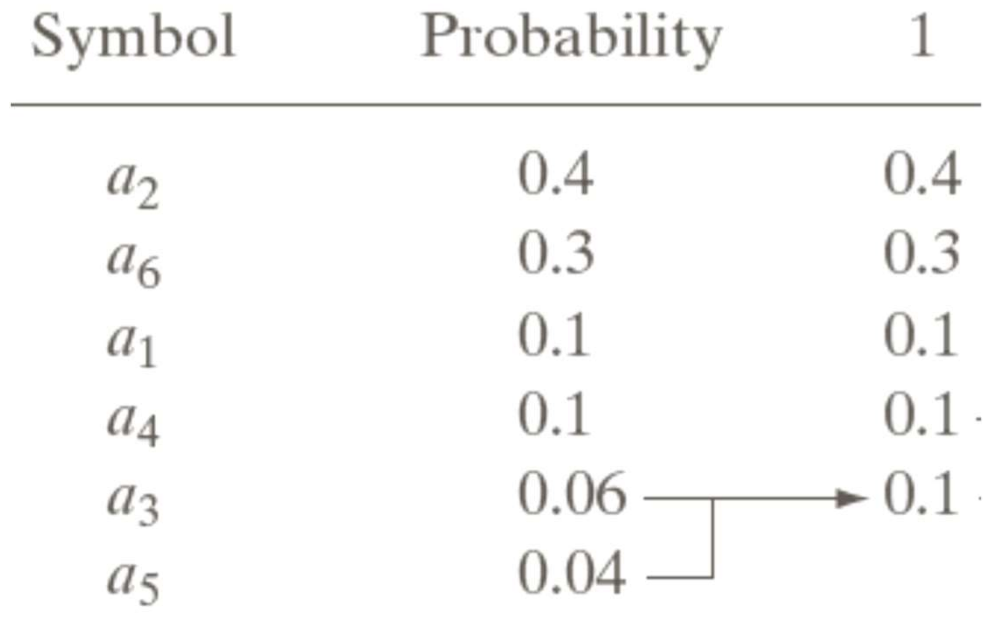
- **Coding procedure:**
- Let a set of six source symbols with probabilities given below are to be coded.
- $\{a_1, a_2, a_3, a_4, a_5, a_6\} = \{0.1, 0.4, 0.06, 0.1, 0.04, 0.3\}$
- Step 1:
- Arrange the probabilities of the given symbols in the descending order. Now, the source symbol becomes



Symbol	Probability
a_2	0.4
a_6	0.3
a_1	0.1
a_4	0.1
a_3	0.06
a_5	0.04

Image Compression

- Step 2:
- Source reduction is created by adding the lowest two probabilities into a single symbol.
- This symbol is known as the compound symbol.
- Then, the two probabilities are replaced by the compound symbol and its probability in the next source reduction

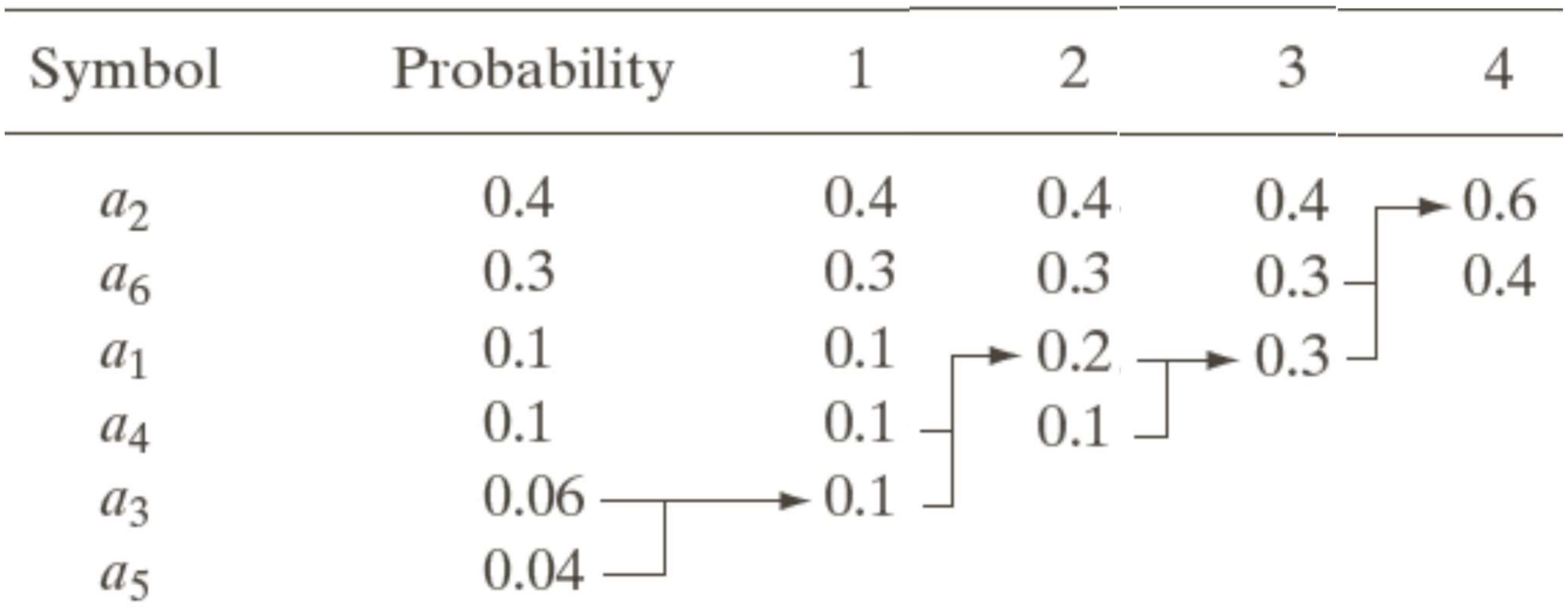


Symbol	Probability	1
a_2	0.4	0.4
a_6	0.3	0.3
a_1	0.1	0.1
a_4	0.1	0.1
a_3	0.06	0.1
a_5	0.04	

Image Compression

- Steps 1 and 2 are repeated until a source with only two symbols is obtained.
- In the third source reduction step, the lowest two probabilities are 0.3 and 0.3.
- These two are added to form a compound symbol with probability 0.6. thus, the probabilities are replaced by 0.6 in the fourth reduction step.
- At last they are ordered in the decreasing order.
- These steps are illustrated in below for the given source symbols.
- The result will be as in the following table

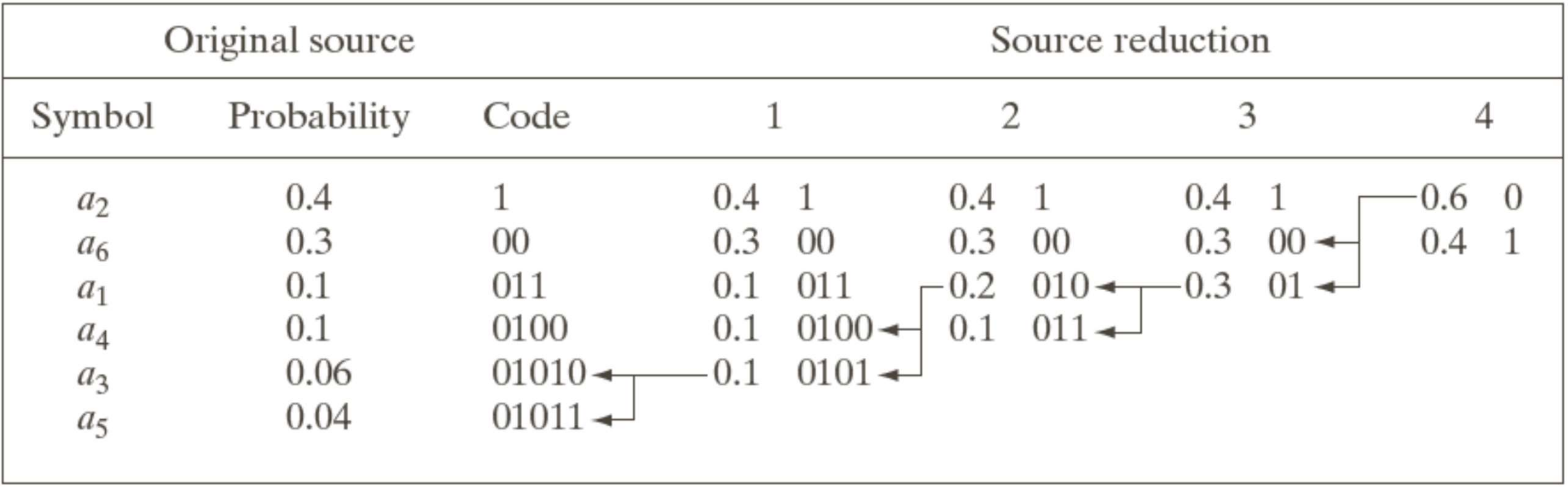
Image Compression



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				

Image Compression

- Step 3: In this step, each reduced source is coded.
- It starts from the smallest source obtained in the last step and goes back to the original source.
- The minimal length binary codes used are: 0 and 1



Original source			Source reduction							
Symbol	Probability	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								

Image Compression

- Here, the reduced symbols 0.6 and 0.4 in the last column are assigned 0 and 1 first.
- Since 0.6 was generated by adding 0.3 and 0.3 in the third column a ‘0’ and ‘1’ are appended.
- Then a ‘0’ and ‘1’ are appended with 01 since its symbol 0.3 was generated by adding 0.2 and 0.1 in the second column. This produces the codes 010 and 011.
- This procedure is repeated until it reaches the original symbols.

Image Compression

- Huffman’s code creates optimal code for a set of symbols with the constraint that, symbols to be coded one at a time
- After the code is generated, coding and/or error-free decoding is done in a simple look up table manner
- The code is an instantaneous uniquely decodable block code.
- It is called as block code because each source symbol is mapped to a fixed sequence of code symbols
- It is instantaneous because, each code word in a string of code symbols can be decoded without referencing succeeding symbols
- It is uniquely decodable because any string of code symbols can be decoded in only one way

Image Compression

- Arithmetic coding:
- These are non-block codes
- In these one-to-one correspondence between source symbols and code words do not exist.
- Arithmetic coding is a data compression technique that encodes data (character, image,….) by creating a code which represents a fraction in the unit interval [0, 1].
- The algorithm is recursive.
- And on each recursion, the algorithm successively partitions subintervals of the unit interval [0, 1].
- This means in arithmetic coding, instead of using a sequence of bits to represent a symbol, a subinterval of the unit interval [0, 1] is used to represent that symbol.

Image Compression

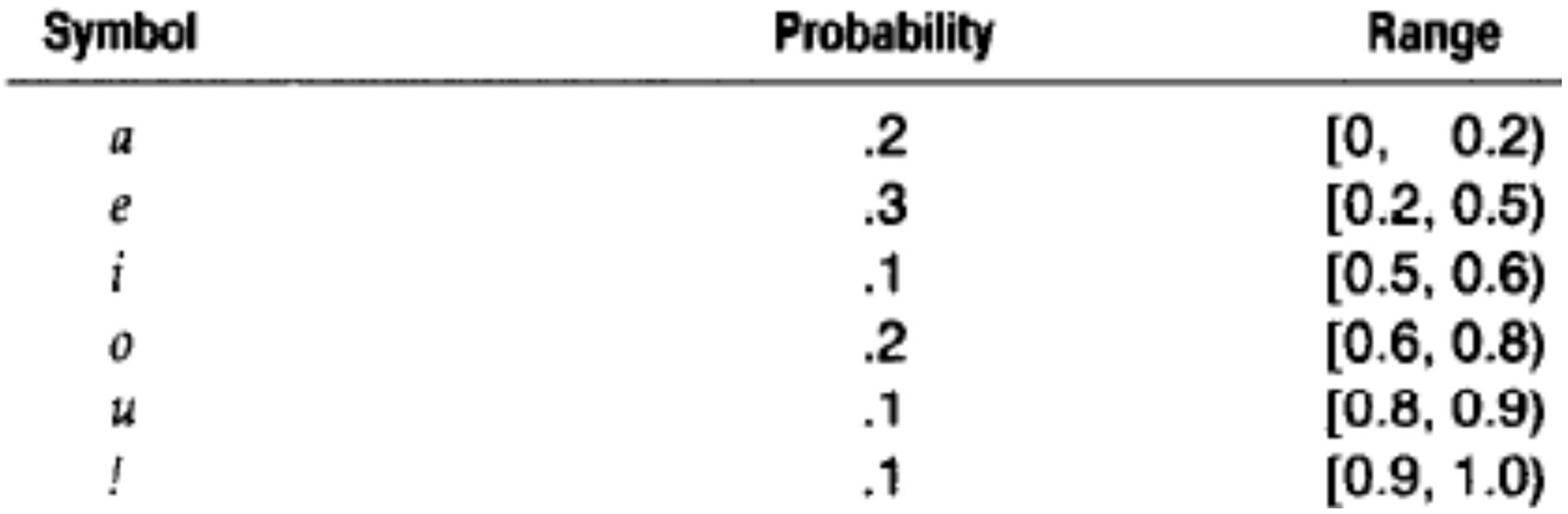
- In other words, the data is encoded into a number in the unit interval[0, 1].
- This technique can be implemented by separating the unit interval into several segments according to the number of distinct symbols.
- The length of each segment is proportional to the probability of each symbol, and then the output data is located in the corresponding segment according to the input symbol

Image Compression

- Arithmetic coding:
- Here a message is represented by an interval of real numbers between 0 & 1.
- As the message becomes longer, the interval needed to represent it becomes smaller, and the number of bits needed to specify that interval grows.
- Successive symbols of the message reduce the size of the interval in accordance with the symbol probabilities generated by the model.
- The more likely symbols reduce the range by less than the unlikely symbols
- Hence add fewer bits to the message.
- Before anything is transmitted, the range for the message is the entire interval [0, l), denoting the half-open interval $0 \leq x < 1$.
- As each symbol is processed, the range is narrowed to that portion of it allocated to the symbol.

Image Compression

- E.g.: suppose the alphabet is (a, e, i, o, u, !), and a fixed model is used with probabilities as shown in Table below.

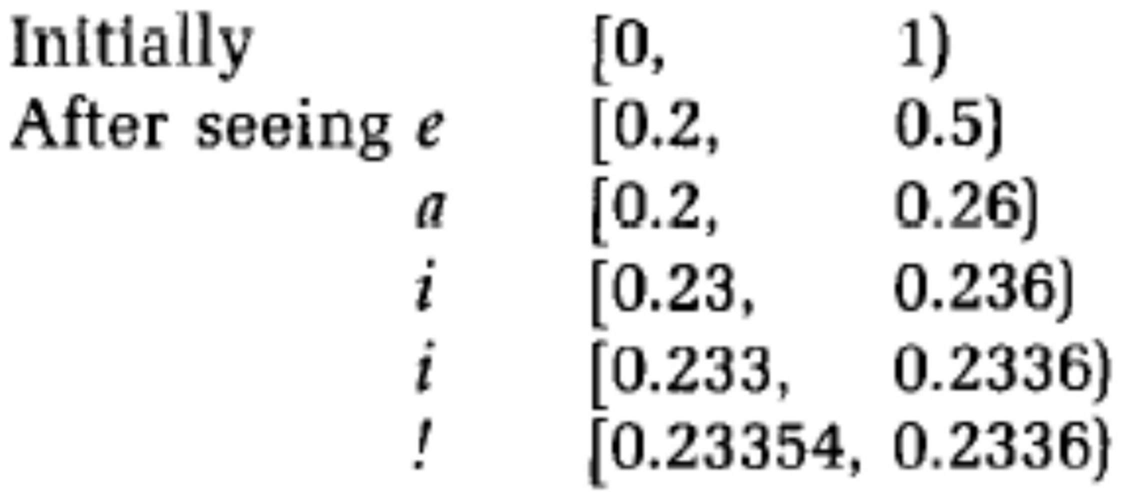


Symbol	Probability	Range
a	.2	[0, 0.2)
e	.3	[0.2, 0.5)
i	.1	[0.5, 0.6)
o	.2	[0.6, 0.8)
u	.1	[0.8, 0.9)
!	.1	[0.9, 1.0)

- Let us assume that transmitted message is eaii!
- Initially both encoder and decoder know the interval as [0,1)

Image Compression

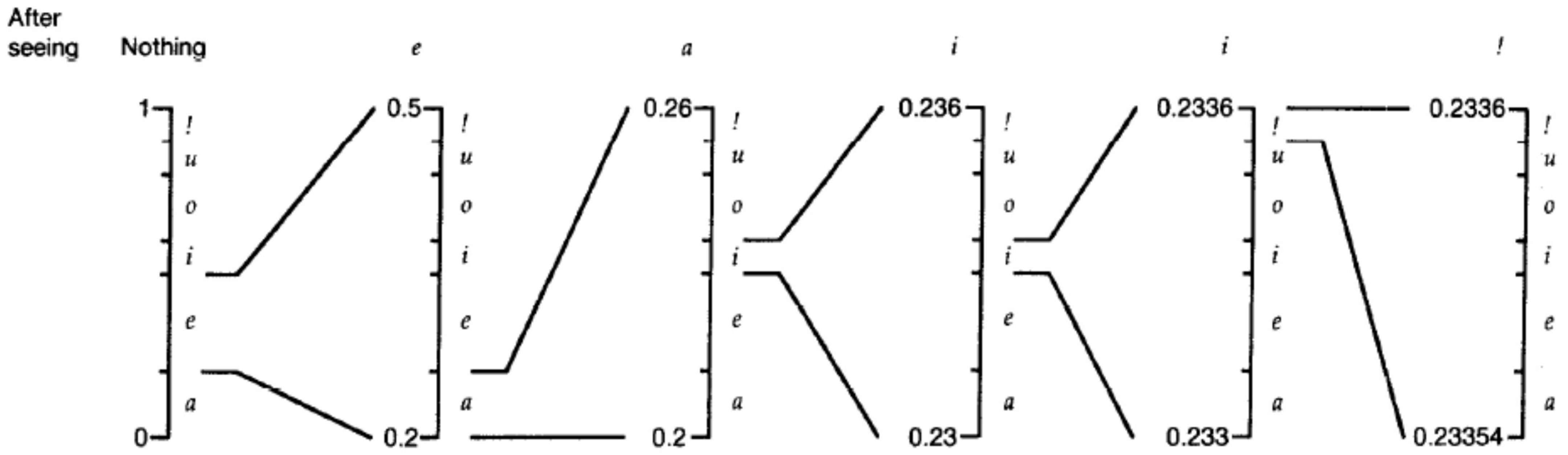
- When the first symbol e arrives at encoder, it narrows the range to [0.2,0,5)
- The second symbol a, when arrives at encoder, further narrows the new range to first one-fifth of this range
- This is because, model had allocated range of [0, 0.2).
- Thus the new range will be [0.2, 0.26)
- Proceeding this way, we get encoded message as shown below



Initially	[0,	1)
After seeing *e*	[0.2,	0.5)
a	[0.2,	0.26)
i	[0.23,	0.236)
i	[0.233,	0.2336)
!	[0.23354,	0.2336)

Image Compression

- Schematically we can represent this as shown below



- This shows the ranges expanded to full height at every stage and marked with a scale that gives end points as numbers

Image Compression

- Lempel-Ziv-Welch (LZW) Coding
- Previous coding techniques focused on removal of coding redundancy
- LZW is an error-free compression technique that also removes spatial redundancy
- This method assigns fixed length code words to variable length sequence of source symbols
- Key feature- it does not require any prior knowledge of probability of occurrence of symbols to be encoded

Image Compression

- An Example
- Consider an image of size 512*512, 8 bit image.
- Uncompressed TIFF Version of this image requires 286,740 bytes of disk space
- Using TIFF's LZW compression option, however, the resulting file is 224,420 bytes.
- The compression ratio ,C= (286740 / 224420)= 1.277
- For Huffman encoded representation of the same image achieves compression ratio, C=1.077.
- The additional compression realized by LZW approach is due to <u>removal of Image's spatial redundancy</u>

Image Compression

- Basic Principles of LZW Coding
- LZW coding assigns fixed length code words to variable length sequence of input symbols.
- The coding is based on a “dictionary” or “codebook” containing the source symbols to be encoded.
- The coding starts with an initial dictionary, which is enlarged with the arrival of new symbol sequences.
- For 8-bit image first 256 words of dictionary are assigned with intensities 0, 1, 2, …255
- As the encoder examines the image pixels, image intensities that are not in the dictionary are placed in unused next location of the dictionary

Image Compression

- E.g.: if the first two pixels of the image are white, sequence 255-255 will be assigned to location 256
- So next time, that two consecutive white pixels are encountered, code word 256 will be used to represent them
- Now suppose if a 9-bit 512-word dictionary is used in the coding process, original 8+8 bits that were used to represent two pixels are now replaced by single 9 bit code word.
- Note that the size of dictionary is important parameter
- If size is too less, detection of matching intensity level will be less likely
- If it is too large, size of the code words may adversely affect compression
-

Image Compression

- Consider the following image

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

- A 512-word dictionary with initial content as shown below is assumed

Dictionary Location	**Entry**
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

Currently Recognized Sequence	Pixel Being Processed	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		

Image Compression

- Note that locations 256-511 are unused.
- Let us encode pixels in left-to-right and top-to-bottom manner
- Each successive intensity value is concatenated with a variable (column 1 of the table- currently recognized sequence)
- This variable is initially empty.
- Dictionary is searched for each concatenated sequence and if found, it will be replaced by the concatenated and recognized sequence.
- This is done in the first row,

Image Compression

sequence. This was done in column 1 of row 2. No output codes are generated, nor is the dictionary altered. If the concatenated sequence is not found, however, the address of the currently recognized sequence is output as the next encoded value, the concatenated but unrecognized sequence is added to the dictionary, and the currently recognized sequence is initialized to the current pixel value. This occurred in row 2 of the table. The last two columns detail the intensity sequences that are added to the dictionary when scanning the entire 4 × 4 image. Nine additional code words are defined. At the conclusion of coding, the dictionary contains 265 code words and the LZW algorithm has successfully identified several repeating intensity sequences—leveraging them to reduce the original 128-bit image to 90 bits (i.e., 10 9-bit codes). The encoded output is obtained by reading the third column from top to bottom. The resulting compression ratio is 1.42:1. ■

Image Compression

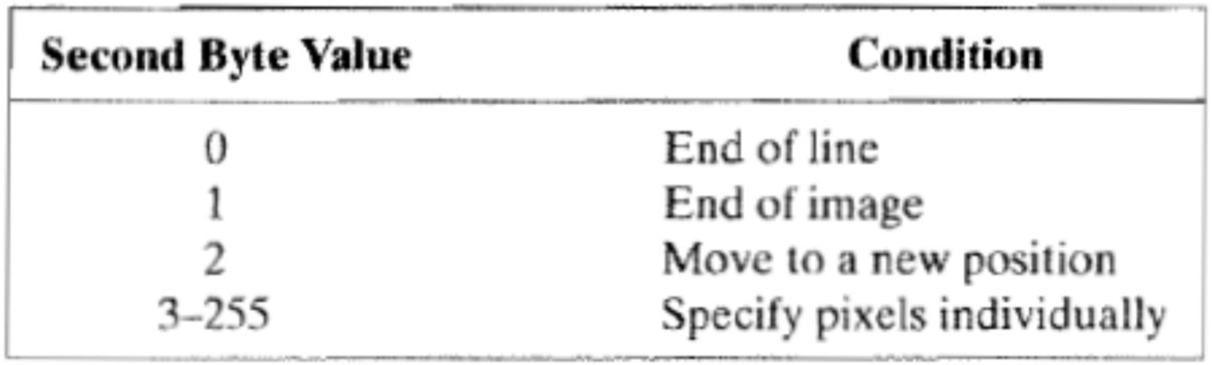
- Run length coding
- In most images we can see that, pixels are correlated spatially in both x and y directions
- This means that, most of the pixel intensities can be predicted from neighbors and hence information carried by single pixel is small.
- Thus there exists more redundancy which can be reduced
- To reduce this, 2-D array needs to be transformed to more efficient non-visual representation
- To do this, run lengths or differences in pixel intensities can be used
- These types of transformation are called as mappings
- Mappings can be reversible or irreversible

Image Compression

- Basic idea
- Repeating intensities along row or column can be compressed by representing runs of identical intensities as run-length pairs
- Each run-length pair represents the beginning of new intensity and number of pixels having that intensity
- BMP file format uses RLE in which image data is represented in two modes
    - Encoded and absolute
- In encoded mode, two byte RLE representation is used
- First byte specifies the no. of consecutive pixels whose intensity level is specified in second byte
- For gray scale images, these will be one of 256 values (0 to 255)

Image Compression

- In absolute mode, first byte is zero and second byte signals one of the four conditions as shown below



Second Byte Value	Condition
0	End of line
1	End of image
2	Move to a new position
3–255	Specify pixels individually

Image Compression

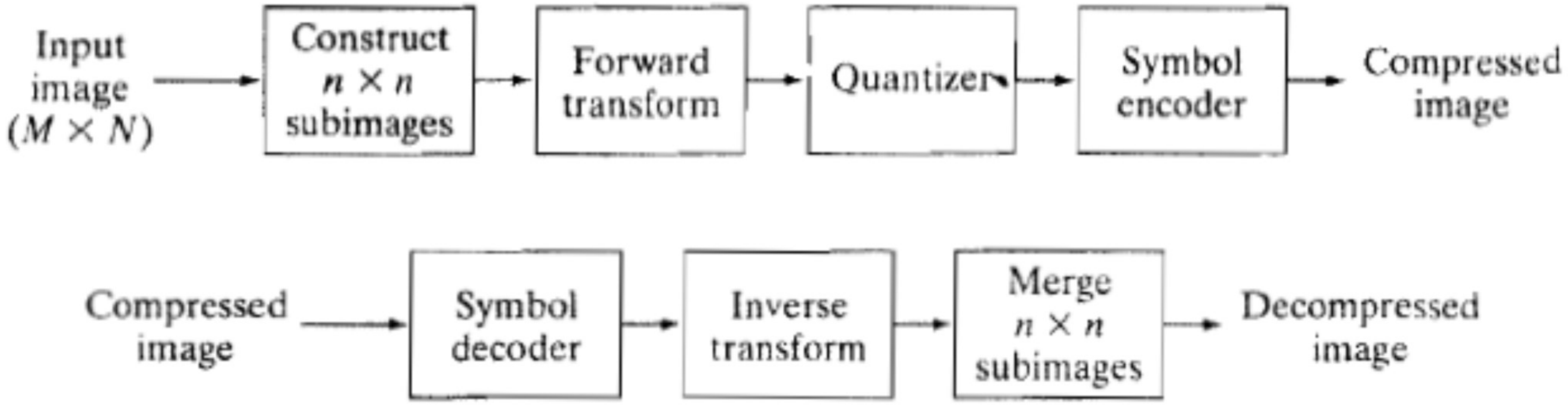
- RLE more efficient for binary images
- Since there are only two intensities, adjacent pixels are more likely to be identical
- Here each row can be represented by a sequence of lengths only instead of run-length pairs

Image Compression

- Transform Coding
- The coding techniques discussed so far operate directly on the pixels of an Image and thus are spatial domain methods.
- Here we study compression techniques that are based on modifying the transform of an image.
- In transform coding, a reversible, linear transform (such as the Fourier transform) is used to map the image into a set of transform coefficients, which are then quantized and coded.
- For most natural images, a significant number of coefficients will have small magnitudes and can be coarsely quantized (or discarded entirely) with little image distortion.
- A variety of transformations, can be used to transform - Discrete Fourier transform (DFT), Wavelet transforms etc

Image Compression

- Figure below shows a typical transform coding system.



- The encoder, performs four straightforward operations - subimage decomposition, transformation, quantization. and coding

Image Compression

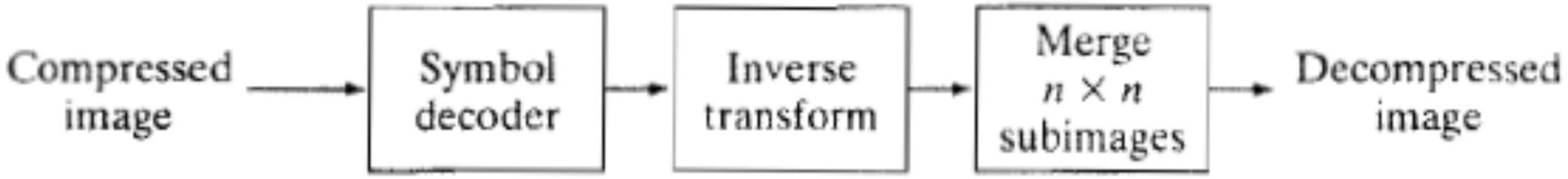
- n x n subimages are converted into transform arrays.
- This tends to decorrelate pixel values and pack as much information as possible in the smallest number of coefficients.
- Quantizer selectively eliminates or coarsely quantizes the coefficients with least information.
- Symbol encoder uses a variable-length code to encode the quantized coefficients.
- Any of the above steps can be adapted to each subimage (adaptive transform coding), based on local image information, or fixed for all subimages.

Image Compression

- An N X N input image first is subdivided into subimages of size n X n.
- These are then transformed to generate $(N/n)^2$ subimage transform arrays each of size n x n.
- The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients.
- The quantization stage then selectively eliminates quantizes the coefficients that carry the least information.
- These coefficients have the smallest impact on reconstructed subimage quality.
- The encoding process terminates by coding (normally using a variable length code) the quantized coefficients.

Image Compression

- If we choose any or all of the transform encoding steps then it is called adaptive transform coding or
- If we use fixed transform encoding, for all subimages, called non-adaptive transform coding.
- Decoder



- Does the reverse operation of encoder

Image Compression

- Transform selection
- Block transform coding systems based on a variety of discrete 2-D transforms have been constructed and/or studied extensively.
- The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and the computational resources available.
- Compression is achieved during the quantization of the transformed coefficients (not during the transformation step).
- Consider a subimage *g(x, y)* of size n x n whose forward, discrete transform *T(u, v)* can be expressed in terms of the general relation

$$T(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x, y) r(x, y, u, v)$$

- for u,v = 0, 1,2, ... , n-1.

Image Compression

- Given T(u, v), g(x, y) similarly can be obtained using the generalized inverse discrete transform

$$g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v)\, s(x, y, u, v)$$

- for x, y = 0,1,2, ... , n-1.
- In these equations, r(x, y, u, v) and s(x, y, u, v) are called the forward and inverse transformation kernels respectively
- they also are referred to as basis functions or basis images
- The T(u, v) for u, v = 0,1, 2,. .. , n-1 are called transform coefficients
- They can be viewed as the expansion coefficients of a series expansion of f(x, y) with respect to basis functions s(x, y, u, v).

Image Compression

- Examples of transform functions
- Walsh-Hadamard Transform
- Discrete Cosine Transform
- Discrete Fourier Transform

Image Compression

- Subimage size selection
- Another significant factor affecting transform coding error and computational complexity is subimage size.
- In most applications. images are subdivided so that the correlation (redundancy) between adjacent subimages is reduced to some acceptable level and so that n is an integer power of 2 where, n is the subimage dimension.
- In general, both the level of compression and computational complexity increase as the subimage size increases.
- The most popular subimage sizes are 8 X 8 and 16 x 16.

Image Compression

- Continued..