Unit-3

Image compression –Chapter 8

Fundamentals, Basic methods, Digital image watermarking, Full motion video compression.

Image Compression- Fundamental

- The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information.
- Data and information are not synonymous- In fact, data are the means by which information is conveyed. Various amounts of data may be used to represent the same amount of information.
- Representations that contain repeated or irrelevant information are said to contain redundant data.
- Data redundancy is a central issue in digital image compression.

Data Redundancy

- Data redundancy is not an abstract concept but a mathematically quantifiable entity.
- Let n_1 and n_2 denote the number of bits in two representations of the same information, the relative data redundancy R of the representation with n_1 bits is defined as-

$$R = 1 - \frac{1}{C}$$ where C is called *compression ratio* is given by, $C = n_1/n_2$

- If C=10 (sometimes written 10:1), the large representation has 10 bits of data for every 1 bit of data in the smaller representation.
- The corresponding relative data redundancy of the larger representation is 0.9 implies that 90% of the data in the first data set is redundant.

Data Redundancy

- In digital image compression, three basic data redundancies can be identified and exploited:
  - Coding redundancy,
  - Interpixel redundancy (Spatial &temporal Redundancy)
  - psychovisual redundancy (irrelevant information)
- Data compression is achieved when one or more of these redundancies are reduced or eliminated.

Coding Redundancy

Coding Redundancy-

- A code is a system of symbols (letters, numbers, bits, etc) used to represent a body of information or set of events.
- Each piece of information of events is assigned a sequence of code symbols called a code word.
- The no of symbols in each code word is its length .
- In this, we utilize formulation to show how the gray-level histogram of an image also can provide a great deal of insight into the construction of codes to reduce the amount of data used to represent it.

Coding Redundancy

- Let us assume, that a discrete random variable r_k in the interval- [0, L-1] represents the intensities of an M X N image and that each r_k occurs with probability $p_r(r_k)$ –

$$p_r(r_k) = n_k/MN \qquad k=0,1,2,\ldots\ldots,L-1\ ,$$

  Where L is the no of intensity values and n_k is the number of times that the kth intensity appears in the image

- If the number of bits used to represent each value of r_k is *$l(r_k)$,* then the *average number of bits required to represent each pixel is* -

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)p_r(r_k)$$

That is, the average length of the code words assigned to the various intensity values is found by summing the product of the number of bits used to represent each intensity and the probability that the intensity occurs.

Thus the total number of bits required to code an M X N image is **MNL_{avg}**

Coding Redundancy

- Fixed length code-
  - If the intensities are represented using a *natural m bit* fixed length code, the right hand side of equation reduces to *m bits.*
  - That is L_{avg}=m when m is substituted for $l(r_k)$.
  - The constant m can be taken outside the summation, leaving only the sum of the $p_r(r_k)$ for $0 \leq k \leq L-1$, which is 1.

Example- variable length coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

- The intensity distribution is shown in second column for an image.
- If a natural 8 bit binary code is used to represent its 4 possible intensities, L_{avg} the average number of bits for code 1 – is 8 bits, because $l(r_k)$ i= 8 for all r_k.
- In code 2 scheme, the average length of the encoded pixel is –

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

The total no of bits needed to represent the entire image is MNL_{avg}=256 X 256 X 1.81=118621

Example variable length coding

- The resulting compression and corresponding relative redundancy are-

$$C = (256 * 256 * 8)/118{,}621 = 8/1.81, \cong 4.42$$

$$\text{And } R = 1 - (1/4.42) = 0.774$$

- Thus 77.4% of the data in the original 8-bit 2-D intensity array is redundant
- The compression achieved by code 2 results from assigning fewer bits to the more probable intensity values than to the less probable one.

Example on Variable-Length Coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

TABL
Exan
varia
codir

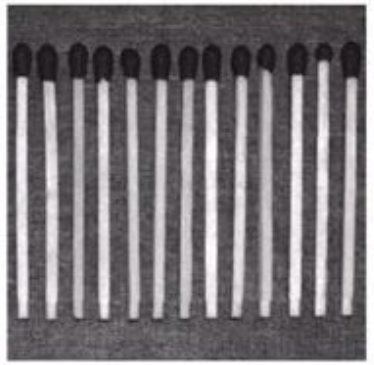
- Average for code 1 is 3, and for code 2 is 2.7
- Compression ratio is 1.11 (3/2.7), and level of reduction is $R_D = 1 - \frac{1}{1.11} = 0.099$

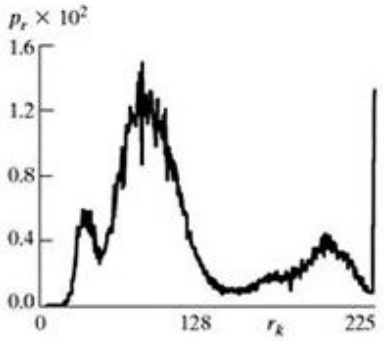
Interpixel - Spatial and temporal redundancy

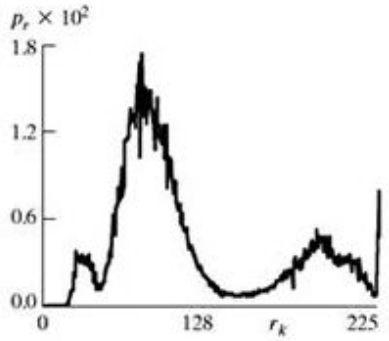
Here the two pictures have Approximately the same Histogram.

We must exploit Pixel Dependencies.

Each pixel can be estimated From its neighbors.







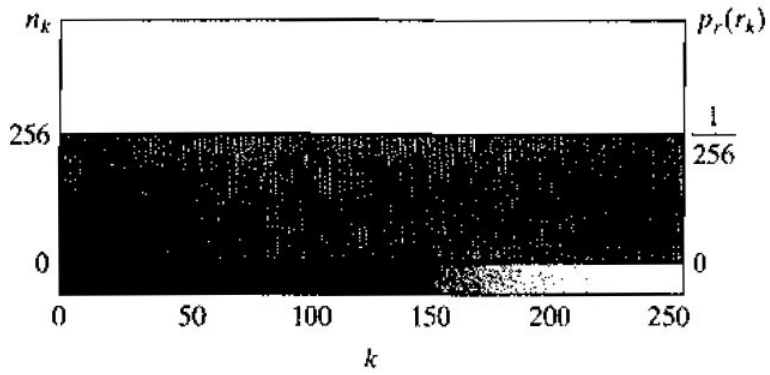
Interpixel - Spatial and temporal redundancy

In fig 8.1b(b)-

- All 256 intensities are equally probable as the histogram of the image is uniform.
- The pixels along each line are identical, they are maximally correlated (depend on each other) in horizontal direction
- We can observe a significant spatial redundancy that can be eliminated by representing the image as a sequence of ***run length pair.***

a b c

FIGURE 8.1 Computer generated 256 × 256 × 8 bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)



Histogram

Run length coding

- What is run length coding?
  - Run-length Encoding, or RLE is a technique used to reduce the size of a repeating string of characters. This repeating string is called a run; typically RLE encodes a run of symbols into two bytes, a count and a symbol.
- Where each Run Length Pairs specifies the start of new intensity and the number of consecutive pixels that have that intensity.
- Each pair consists of:
  - Intensity values
  - Number of pixels that have this intensity value
- Transformation of this type are called *mapping.*

Irrelevant information- psychovisual redundancy

- Informations that are ignored by human visual system or is extraneous to the intended use of an image are obvious candidates for omission.

The human visual system is more sensitive to edges

Middle Picture:
Uniform quantization from 256 to 16 gray levels
$C_R = 2$

Right picture:
Improved gray level quantization (IGS)
$C_R = 2$

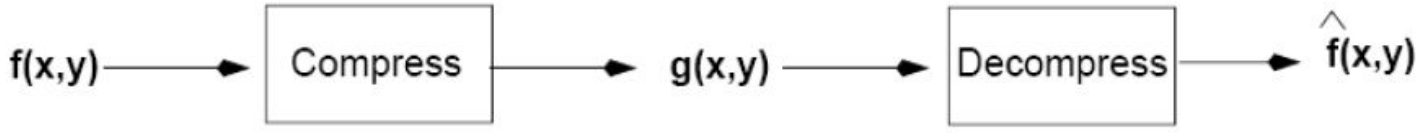
Fidelity Criteria

- The removal of psychovisually redundant data results in a loss of real or quantitative visual information.
- Because information of interest may be lost, a repeatable or reproducible means of quantifying the nature and extent of information loss is highly needed.
- Two general classes of criteria are used as the basis for such an assessment:

1. Objective fidelity criteria
2. Subjective fidelity criteria.

Fidelity Criteria

- When lossy compression techniques are employed, the decompressed image will not be identical to the original image. In such cases, we can define **fidelity criteria** that measure the difference between this two images.



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

- Objective fidelity criteria-
  - When the level of information loss can be expressed as a mathematical function of the original or input image and the compressed and subsequently decompressed output image, it is said to be based on an objective fidelity criterion.
  - A good example is the root-mean-square (rms) error between an input and output image.

Objective fidelity criteria

- Images are of size M * N.
- The e_{rms} between two images is then the *square root of the squared error averaged* over the M*N array.

- $f(x,y)$ is the input image
- $\hat{f}(x,y)$ is the estimate or approximation of $f(x,y)$ resulting from compression and decompression
- Error between the two images

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

- Root-mean square error:

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

Objective fidelity criteria

If $\hat{f}$ (x,y) is considered to be the sum of original image f(x,y) and an error or noise signal e(x,y), the mean square signal to noise ratio of the output image is SNR_{ms} is defined as -

$$\mathrm{SNR}_{\mathrm{ms}} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}$$

Subjective fidelity criteria

- Although objective fidelity criteria offer a simple and convenient mechanism for evaluating information loss, most decompressed images ultimately are viewed by humans.
- Consequently, measuring image quality by the subjective evaluations of a human observer often is more appropriate.

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.

Image compression models

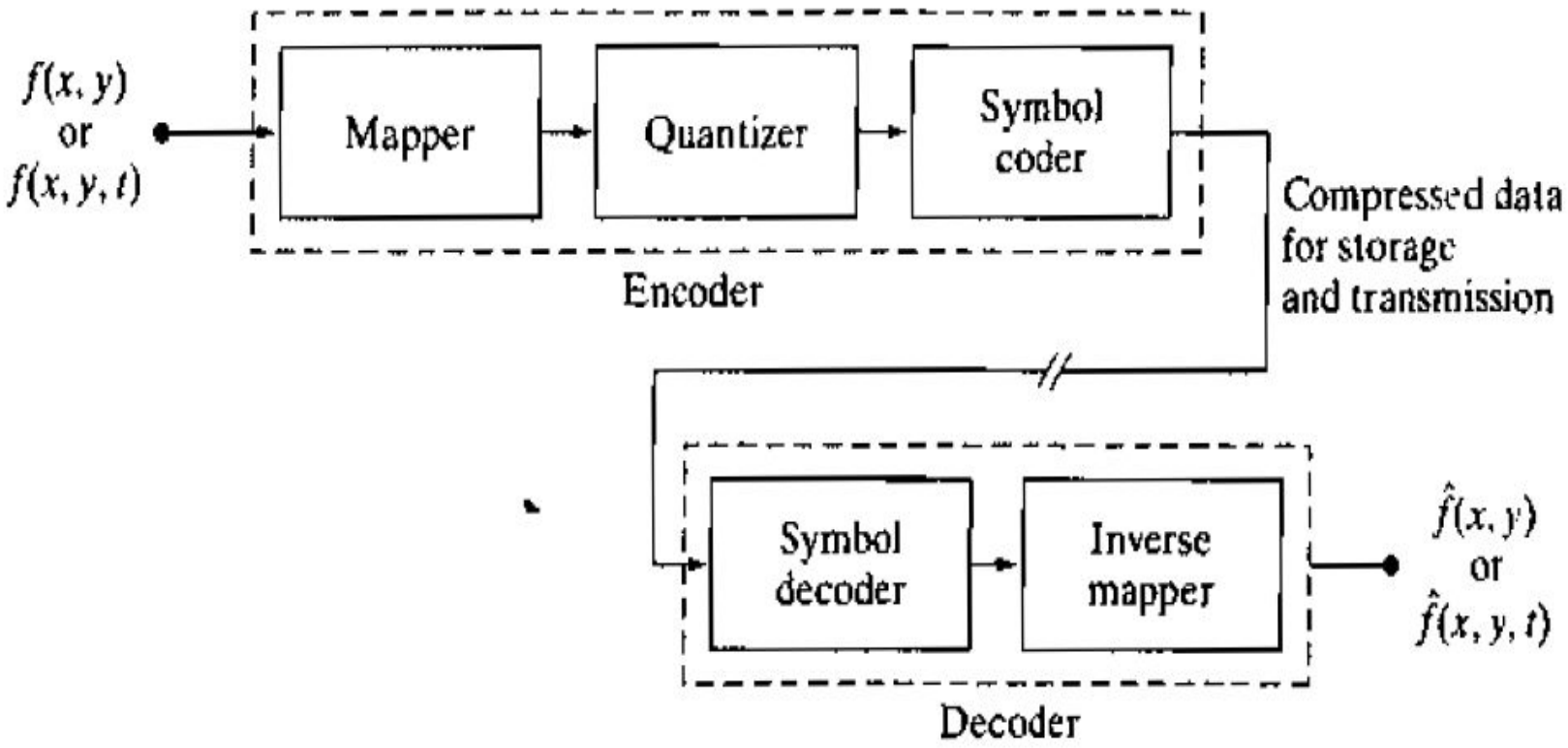


FIGURE 8.5 Functional block diagram of a general image compression system.

A compression system consists of two distinct structural blocks: *an encoder and a decoder.*

Image compression models

- *An encoder –*
  - Encoder is responsible for reducing or eliminating any coding, interpixel or psychovisual redundancy.
  - The first block “Mapper” transforms the input data into a nonvisual format, designed to reduce interpixel redundancy. This operation generally is reversible and may or may not reduce directly the amount of data required to represent the image. For eg, run-length encoding
  - The quantizer reduces accuracy of the mapper output in accordance with some fidelity criterion. This stage reduces the psychovisual redundancies of the input image. This operation is irreversible. Thus it must be omitted when error-free compression is desired.
  - The symbol encoder creates a fixed or variable length codeword. The operation, of course, is reversible.
  - So, The source encoder is responsible for reducing or eliminating any coding, interpixel, or psychovisual redundancies in the input image.

Image compression models

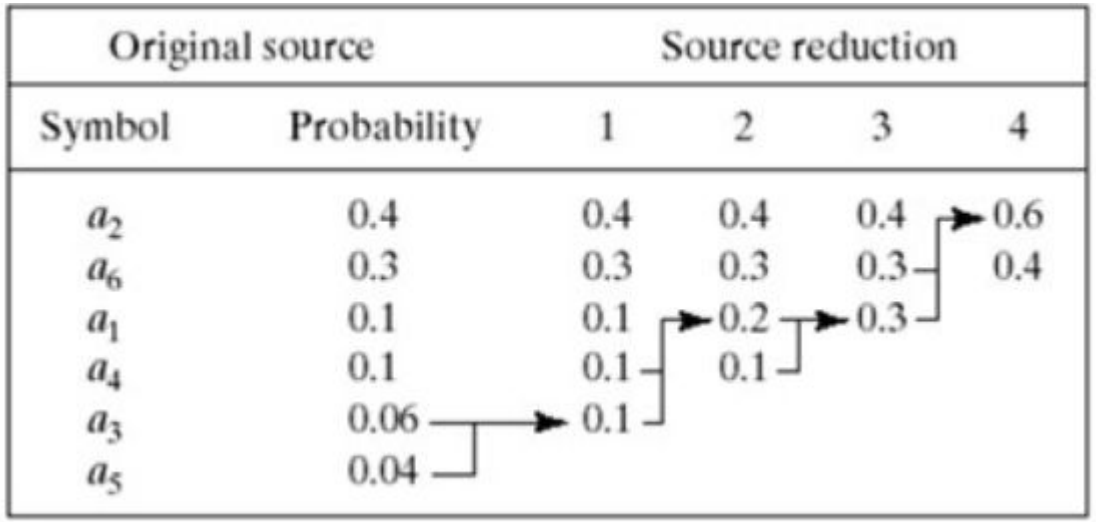
- *A decoder-*
- The source decoder shown contains only two components: a symbol decoder and an inverse mapper.
- These blocks perform, in reverse order, the inverse operations of the source encoder's symbol encoder and mapper blocks.
- Because quantization results in irreversible information loss, an inverse quantizer block is not included in the general source decoder model

Compression Method- Huffman Coding

- The most popular technique for removing coding redundancy is due to *Huffman*.
- When coding the symbols of an information source individually, Huffman coding yields the *smallest possible number of code symbols* per source symbol.
- The first step in Huffman's approach is to create a series of source reductions by ordering the probabilities of the symbols under consideration and combining the lowest probability symbols into a single symbol that replaces them in the next source reduction.

Huffman Coding – First step

- At the far left, a hypothetical set of source symbols and their probabilities are ordered from top to bottom in terms of decreasing probability values.
- To form the first source reduction, the bottom two probabilities, 0.06 and 0.04, are combined to form a "compound symbol" with probability 0.1.
- This compound symbol and its associated probability are placed in the first source reduction column so that the probabilities of the reduced source are also ordered from the most to the least probable.

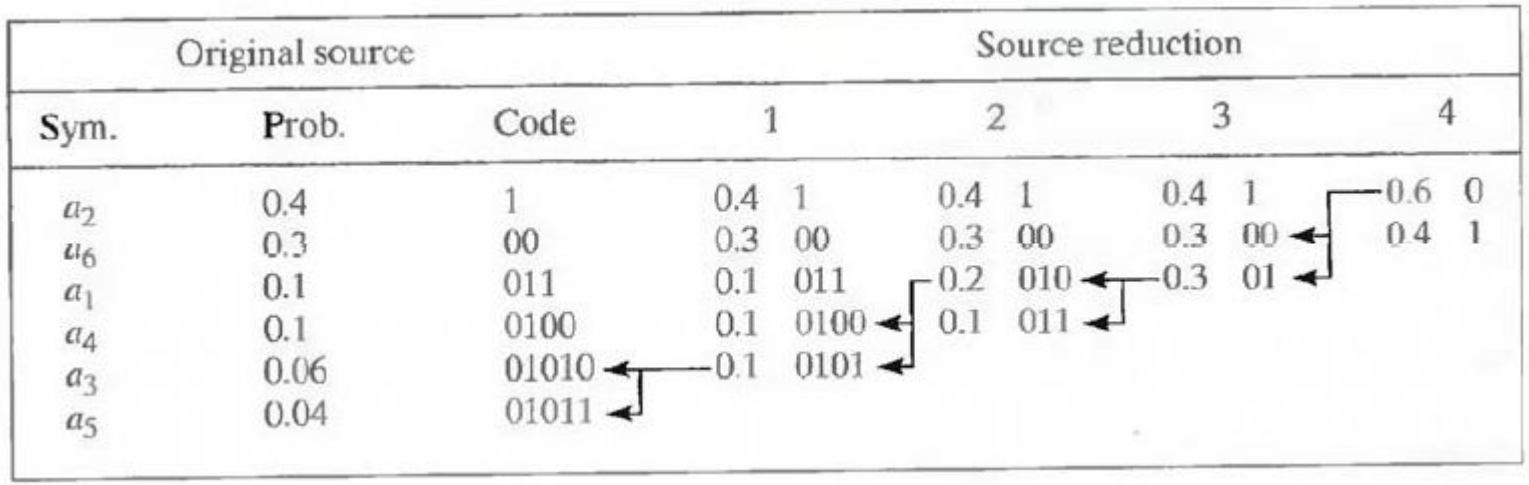


Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

This process is then repeated until a reduced source with two symbols (at the far right) is reached.

Huffman Coding – Second Step

- The second step in Huffman's procedure is to code each reduced source starting with the smallest source and working back t the original source.



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								

- The minimal length binary code for a two-symbol source, is the symbols 0 and 1.
- These symbols are assigned to the two symbols on the right (the assignment is arbitrary; reversing the order of the 0 and 1 would work just as well).
- As the reduced source symbol with probability 0.6 was generated by combining two symbols in the reduced source to its left, the 0 used to code it is now assigned to both of these symbols, and a 0 and 1 are arbitrarily appended to each to distinguish them from each other.
- This operation is then repeated for each reduced source until the original source is reached.

Huffman Coding

- The average length of this code is-

$$L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2 \text{ bits/symbol}$$

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								

LZW Compression Coding

- Its an error free compression technique that also address spatial redundancies in an image.
- The technique, called Lempel-Ziv-Welch (LZW) coding, assigns fixed-length code words to variable length sequences of source symbols.
- It does not requires priori knowledge of the probability of occurrence of the symbols to be encoded.
- LZW compression has been integrated into a variety of mainstream imaging file formats, including the graphic interchange format (GIF), tagged image file format (TIFF), and the portable document format (PDF).
- https://youtu.be/2FjOJMelZe0

LZW Compression Coding

- At the onset of the coding process, a codebook or "dictionary" containing the source symbols to be coded is constructed.
- Dictionary is adaptive to the data.
- For 8-bit monochrome images, the first 256 words of the dictionary are assigned to the gray values 0, 1, 2..., and 255.
- LZW encoder sequentially examine the image's pixels, Gray level sequence that are not in the dictionary are placed in algorithmically determined (e.g., the next unused) locations.

Dictionary Location (Code word)	Entry
0	0
1	1
2	2
. . .	. . .
255	255
256	127-127
...	...
...	...

LZW Compression Coding

- If the first two pixels of the image are white, for instance, sequence “255- 255” might be assigned to location 256, the address following the locations reserved for gray levels 0 through 255.
- The next time that two consecutive white pixels are encountered, code word 256, the address of the location containing sequence 255-255, is used to represent them.
- If a 9-bit, 512-word dictionary is employed in the coding process, the original (8 + 8) bits that were used to represent the two pixels are replaced by a single 9-bit code word.

Dictionary Location (Code word)	Entry
0	0
1	1
2	2
. . .	. . .
255	255
256	127-127
...	...
...	...

Example – LZW coding

- Consider the following 4 x 4, 8-bit image of a vertical edge:
- A 512-word dictionary with the following starting content is assumed:
- Locations 256 through 511 are initially unused.
- The image is encoded by processing its pixels in a left-to-right, top-to-bottom manner.

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

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

Example – LZW coding

- Each successive gray-level value is concatenated with a variable—column 1 of Table called the *"currently recognized sequence."*
- As can be seen, this variable is initially *null or empty.*
- The dictionary is searched for each concatenated sequence and if found, as was the case in the first row of the table, is replaced by the newly concatenated and recognized (i.e., located in the dictionary) sequence. This was done in column 1 of row 2.

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

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		

Digital image watermarking

- One way to discourage illegal duplication is to insert one or more items of information, collectively called a watermark.
  - Watermarking is done in such a way that the watermarks are inseparable from the image themselves.
- Watermarking is presented when a pattern is inserted in an image, video or audio file, it helps to copyright the information in the files.
- Watermark can be visible or invisible

Digital image watermarking

- Visible watermark: a visible watermark is an opaque or semi transparent sub image or image that is placed on top of another image.
  - Television network often place visible watermarks in the upper or lower right hand corner

- A watermark logo can be viewed as the property of the ownership.

Digital image watermarking

- Letting f_w denote the watermarked image, we can express it as a linear combination of the unmarked image f and watermark w using

$$f_w=(1-\alpha)f + \alpha w$$

Where constant α controls the relative visibility of the watermark and the underlying image.

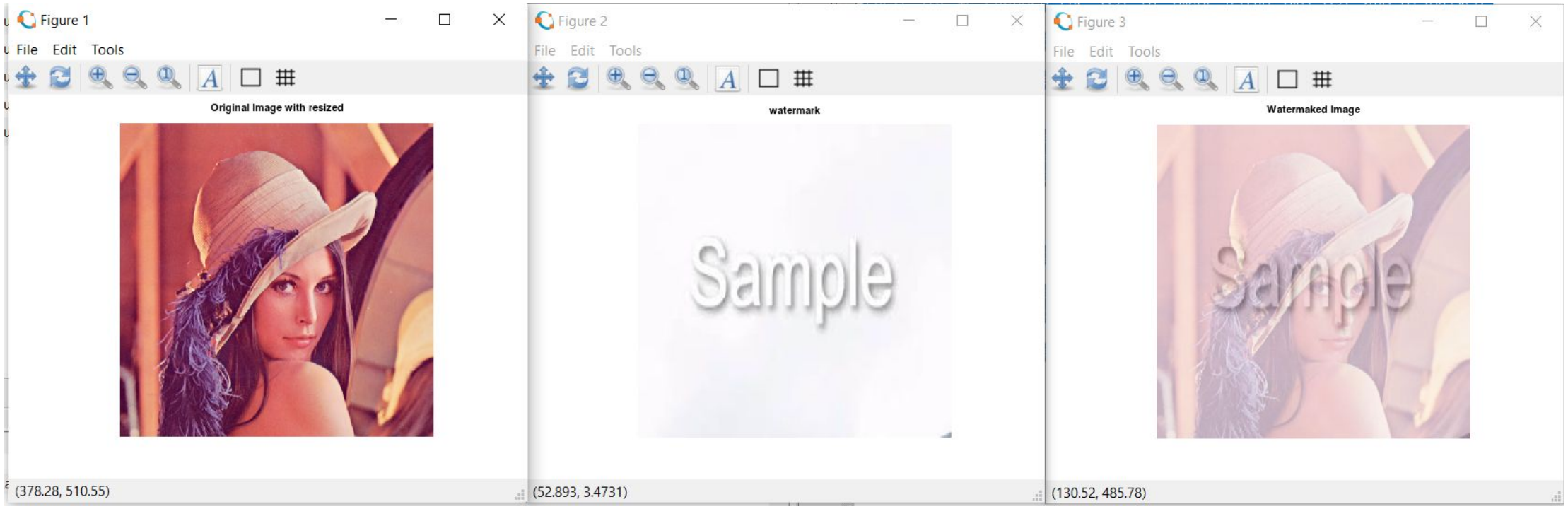
If α is 1, the watermark is opaque and the underlying image is completely obscured.

As α approaches zero, more of the underlying image and less of the watermark is seen.

Digital image watermarking

- pkg load image; clear all; close all;
- #Input Image where we want to apply watermark
- f=imread('lena_color_512.tif');
- #For watermarking, size of inputimage and watermarking image should be same #there for we changed the size of image using imresize and dispalyed
- fr=imresize(f,[560 560]);
- figure;imshow(fr); title('Original Image with resized');
- #Watermarking Image
- w=imread('watersample.jpg');
- wr=imresize(w,[560 560]);
- figure;imshow(wr); title('watermark');
- #Applied watermarking
- alpha=0.7;
- fw=(1-alpha)*fr + alpha.*wr;
- #Display the watermarked Image
- figure;imshow(fw); title('Watermaked Image');

Watermarking -Output



Huff-man coding

- pkg load communications
- sig = repmat([3 3 1 3 3 2 3 3 2 3],1,3);
- symbols = [1 2 3]; p = [0.1 0.2 0.7];
- dict = huffmandict(symbols,p);
- hcode = huffmanenco(sig,dict);
- dhsig = huffmandeco(hcode,dict)

- isequal(sig,dhsig)

- binarySig = de2bi(sig);
- seqLen = numel(binarySig)
- binaryhcode = de2bi(hcode);
- encodedLen = numel(binaryhcode)

```
ans = 1
seqLen = 60
encodedLen = 39
>>
```