Chapter 8: Image Compression

Origin42

Question 1. [Marks: 18]

b) Consider the simple 3x3, 8 bit image as [(3,4,7),(3,4,7),(3,4,7)]. Suppose you want to compress the image using a loss less Lempel-Ziv-Welch (LZW) fixed length coding algorithm. [9]

i. Generate your new codebook using 9 bits and illustrate your step by step LZW encoding process for the above image. [5]

ii. What kind of redundancy of image data it reduces in your encoding process? Explain. [2]

iii. Any compression achieved by employing LZW in your above encoding process? Proof it. [2]

1.b. .i. Solution: 024

Given image matrix,

3	4	7
3	4	7
3	4	7

For 8-bit image, codebook will be 9-bit,

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

Using LZW encoding process,

Current Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Codeword)	Dictionary Entry
	3			
3	4	3	256	3-4
4	7	4	257	4-7
7	3	7	258	7-3
3	4	-	-	-
3-4	7	256	259	3-4-7
7	3	-	-	-
7-3	4	258	260	7-3-4
4	7	-	-	-
4-7		257		

So, new codebook,

Dictionary Location	Dictionary Entry
0 1 ... 255	0 1 ... 255
256	3-4
257	4-7
258	7-3
259	3-4-7
260	7-3-4

So, the output sequence is 3-4-7-256-258-257

1.b. .ii. Solution: 024
In my LZW coding process, it reduces spatial data redundancy. It assigns fixed-length code words to variable length sequences of source symbols.

1.b. .iii. Solution: 024
Before compression,
Total matrix value 9
Each matrix value 8 bits
So, original image size = 9x8 =72 bits

After compression,
Total encoded output 6
Each encoded output 9 bits
So, compressed image size = 6x9 =54 bits

Compression ratio = 72/54 = 1.33 : 1
Yes, compression achieved.

Question 5. [Marks: 13]

a) i. Why image compression is needed? [1] **[5]**
ii. How many and what types of data redundancies are there in an image? [2]
iii. A 512 X 512 8-bit image with 5.3 bits/pixel entropy is to be Huffman Coding. What is the maximum compression that can be expected? [2]

5.a. . Solution: 024

(i)

Image compression is needed for:

- Data store at low storage
- Data transmission at minimum cost

(ii)

In image there are 3 types of data redundancies.
Coding redundancy: Most 2-D intensity arrays contain more bits than are needed to represent the intensities.
Spatial and temporal redundancy: Pixels of most 2-D intensity arrays are correlated spatially and video sequences are temporally correlated.
Irrelevant information: Most 2-D intensity arrays contain information that is ignored by the human visual system.

(iii)

Entropy = 5.3 bits/pixel, means minimum 5.3 bits required to represent a pixel without losing any data.

Compression ratio = (512x512x8) / (512x512x5.3) = 1.509 : 1
This is the maximum compression that can be expected.

b) Consider the simple 3x3, 8 bit image [(18,16,16),(11,11,14),(11,14,16)]. Suppose you want to compress the image using Huffman code algorithm. [8]

i. Illustrate your Huffman encoding process for the above image. [3]
ii. Compute the entropy of the above image. [2]
iii. What kind of redundancy of image data it reduces in your encoding process? Is it a loss less compression technique? [1]
iv. Any compression achieved by employing Huffman coding in your above encoding process? Proof it. [2]

5.b. Solution: 024

(i)

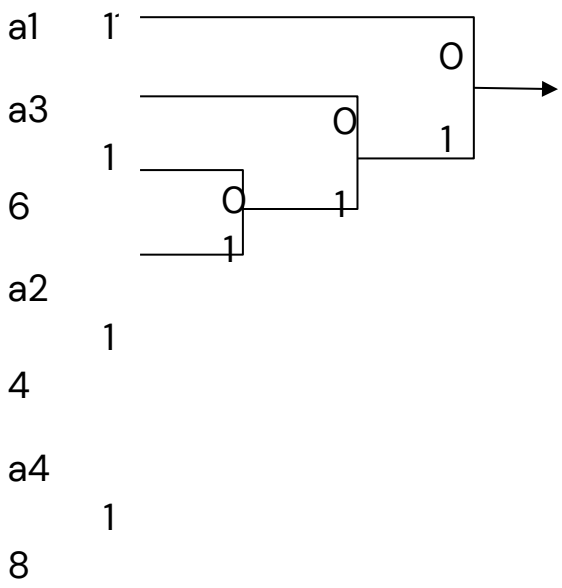
Given image matrix,

18	16	16
11	11	14
11	14	16

PDF,

Symbol	Pixel Value	Probability
a1	11	3/9 = 0.33
a2	14	2/9 = 0.22
a3	16	3/9 = 0.33
a4	18	1/9 = 0.11

Using Huffman coding process,



Symbol	Pixel Value	Codeword
a1	11	0
a3	16	10
a2	14	110
a4	18	111

(ii)

Original image entropy, $H = -[(0.33)x \log_2(0.33) + (0.22)x \log_2(0.22) + (0.33)x \log_2(0.33) + (0.11)x \log_2(0.11)] = 1.7578$ bits/pixel

(iii)

Huffman coding is a Coding Redundancy reduce technique.

Average bit requirement per symbol, Lavg = 0.33x1 + 0.22x3 + 0.33x2 + 0.11x3 = 1.98 bits/pixel

As, H < Lavg. So, this compression is lossless.

(iv)

Compression ratio = 8/1.98 = 4.04 : 1
So, compression was achieved.

Enigma41

Question 2. [Marks: 14]

a) Consider the simple 5×5, 8-bit image as in the 2-D array below: **[8]**

18	16	16	14	12
11	11	12	14	12
11	14	12	12	14
12	16	16	17	17
17	12	11	14	11

i. Derivate the equation to calculate Entropy of an image from the information theory to understand the optimum required bit to represent image information. [2]
ii. Compute the entropy of the above image. [1]
iii. Calculate the respective Huffman codes for each symbol (each pixel value). [2]
iv. What is the compression ratio achieved by employing Huffman coding instead of 8-bit fixed length coding? [2]
v. Calculate the relative data redundancy of the given 8-bit image. [1]

2.a. Solution:

b) i. Supposed an image [1,4,3,2,3,4,2,4,5] of 3x3 of 3 bits is compressed by a lossy compression technique; after de-compression, all the pixel values increased by 2. Calculate the Root Mean Square Error as fidelity criteria. [3] **[6]**

ii. Compute the Golomb code for $G_4(9)$. [3] → 11001

2.b. Solution: 024

(i)

Given, image pixels,

f(x, y)		
1	4	3
2	3	4
2	4	5

=>

f'(x, y)		
3	6	5
4	5	6
4	6	7

Total 9 pixel values. Each pixel value is increased by 2.

Row=M=3, Column=N=3

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

Erms = sqrt((1/9) x (2^2) x 9) = 2

(ii)

Golomb Coding

Given a nonnegative integer n and a positive integer divisor $m > 0$, the Golomb code of n with respect to m, denoted $G_m(n)$, constructed as follows:

Step 1. Form the unary code of quotient $\lfloor n/m \rfloor$
(The unary code of integer q is defined as q 1s followed by a 0)

Step2. Let k=$\lceil \log_2 m \rceil, c = 2^k - m, r = n \bmod m$, and compute truncated remainder r' such that

$$r' = \begin{cases} r \text{ truncated to } k\text{-}1 \text{ bits} & 0 \le r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

Step 3. Concatenate the results of steps 1 and 2.

Gm(n) = G4(9)
m =4
n =9

Step 1:
floor(n/m) = floor(9/4) = 2
So, unary value followed by 0 = 110

Step 2:
k = ceiling($\log_2$(m)) =ceiling($\log_2$(4)) = 2
c =2^k - m = 2^2 - 4 = 0
r = n mod m = 9 mod 4 = 1
r′ = 01 ; [r+c truncated to k bits as 0<=r<c is not accomplished]

Step 3:
Golomb code for G4(9) = " unary value followed by 0 " + " r′ " = 11001

Recursive40

Question 5. [Marks: 14]

a) [8]

i. Why do we need image compression?

ii. How many types of data redundancies are there in an image?

iii. Define a loss less compression technique.

iv. Write down the properties of Arithmetic coding.

v. Consider the Table-1 below that represents the five-symbol sources with the probabilities and initial subintervals. What is the Arithmetic code for the sequence $a_1 a_4 a_3 a_4 a_2 a_5$?

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.3	[0.4, 0.7)
a_4	0.2	[0.7, 0.9)
a_5	0.1	[0.9, 1.0)

5.a. Solution: Rabab 039

(i) Image compression is needed for:

- Low data storage
- Data transmission at lower cost

(ii) There are 3 types of data redundancies: Coding, spatial, irrelevant information

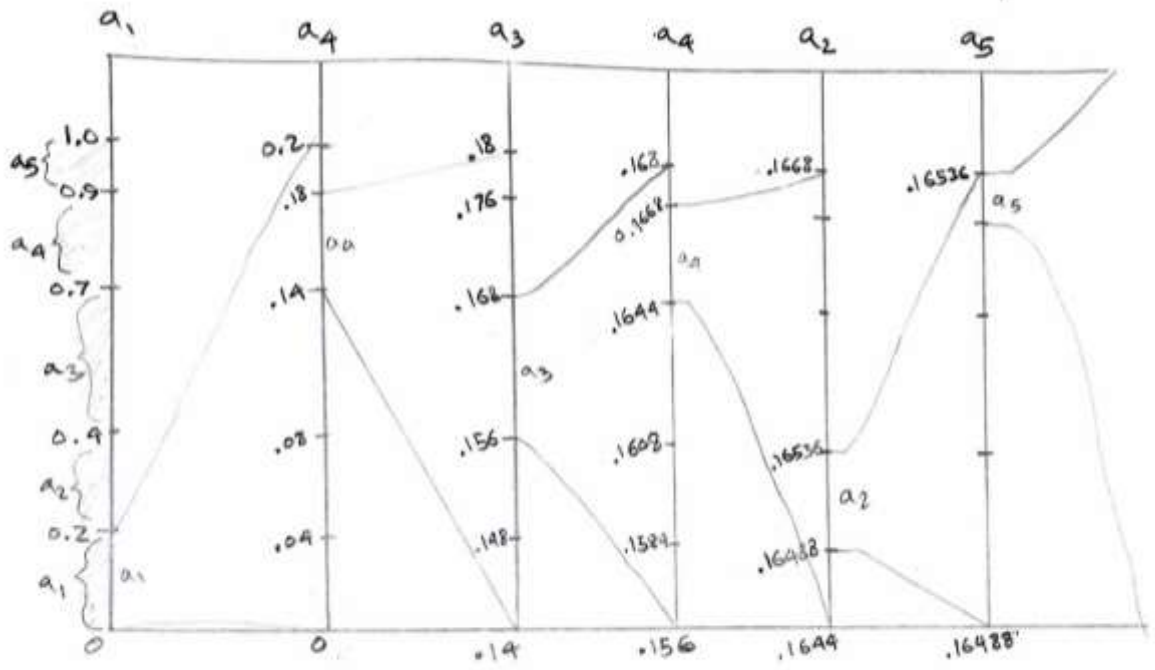
(iii) Arithmetic Coding is a lossless compression technique that encodes data by creating a code string which represents a fractional value on the number line between 0 and 1

(iv)

Loss Less Compression: Arithmetic Coding:

- ***Variable length code***
- ***Error-free compression technique***
- ***Non block coding***
  - one to one correspondence between symbol and code **does not exist**
  - An entire sequence of source symbols (string of symbol) is mapped to a **single arithmetic number (code word)**
  - The code word itself defines an interval of real numbers between 0 and 1.
- **This coding can achieve theoretically higher compression rates than Huffman codes**

(v)



Recursive 40, 5(a) v.

∴ Arithmetic code for the sequence $a_1a_4a_3a_4a_2a_5$ = 0.16536

b) Given a 5x5 pixel image and respective pixel values (8-bit code for each pixel) below, [6]

180	160	160	140	120
110	110	120	140	120
110	140	120	120	140
120	160	160	170	170
170	120	110	140	110

An 8-bit Image

i. Compute the entropy of the image.
ii. Calculate the respective Huffman Codes for each pixel value.
iii. What is the compression ratio achieved by employing Huffman Coding instead of 8-bit fixed length coding?
iv. Calculate the relative data redundancy of the given 8-bit image.
v. Compute the effectiveness of the Huffman coding.

5.b. Solution:

Prototype39

1.

a)	Image compression algorithms are developed by taking advantage of the redundancy that is inherent in image data. How many primary types of redundancies are there in an image and what are those? Discuss one redundancy type that can be overcome using variable length code words. [2+4]	**[6]**

Solution:

b) Given a 5x5 pixel image and respective pixel values (8-bit code for each pixel) below,

180	160	160	140	120
110	110	120	140	120
110	140	120	120	140
120	160	160	170	170
170	120	110	140	110

i. What is Entropy? Give the equation to calculate Entropy. [1+0.5]

ii. Compute the entropy of the image. [1]

iii. Calculate the respective Huffman Codes for each symbol (each pixel value). [2]

iv. What is the compression ratio achieved by employing Huffman Coding instead of 8-bit fixed length coding? [1]

v. Calculate the relative data redundancy of the given 8-bit image. [1]

[6.5]

Solution:

Return38

5.

a) A 1024 X 1024 8-bit image with 5.3 bits/pixel entropy is to be Huffman Coding .What is the maximum compression that can be expected? **[2]**

Solution:

b) Given a 4x8 pixel image and respective pixel values (8-bit code for each pixel) below, **[5]**

8

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

i. Compute the entropy of the image.
ii. Calculate the respective Huffman Codes for each symbol (each pixel value).
iii. What is the compression ratio achieved by employing Huffman Coding instead of 8-bit fixed length coding?
iv. Calculate the relative data redundancy of the given 8-bit image.
v. Compute the effectiveness of the Huffman coding.

Solution: Rabab 039

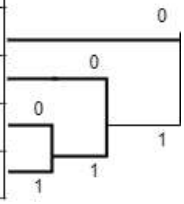
(i) Total pixels in image = 4*8 = 32

Symbol	Pixel value	Probability
a2	95	4/32 = 0.125
a3	169	4/32 = 0.125
a1	21	12/32 = 0.375
a4	243	12/32 = 0.375

Entropy = - [0.125log(0.125) + 0.125log(0.125) + 0.375log(0.375) + 0.375log(0.375) = 0.796 (**Amar 1.81127 ashe ~ ID 077 $\log_2$ hobe**)

(ii) Using Huffman Coding:

Symbol	Pixel value	Probability
a2	95	4/32 = 0.125
a3	169	4/32 = 0.125
a1	21	12/32 = 0.375
a4	243	12/32 = 0.375



Symbol	Pixel value	Probability	Huffman codeword
a2	95	4/32 = 0.125	0

a3	169	4/32 = 0.125	10
a1	21	12/32 = 0.375	110
a4	243	12/32 = 0.375	111

(iii) Lavg = (0.125x1) + (0.125x2) + (0.375x3) + (0.375x3) = 2.625
Compression ratio, C = (4x8x8) / (4x8x2.625) = 8 : 2.625 = 3.0476 : 1

(iv) R_D = 1 - 1/C = 1 - 1/3.0476 = 0.671

(v)

c) i) Suppose we have a grayscale image with most of the values of pixels being same. What can we use to compress the size of the image? **[5]**

In the figure, * is representing the value of last three digits of your Student ID (e.g. for ID 160104001, *** will be 1).**

ii) Show your step by step **LZW** encoding process (to generate your codebook, use **4** bits). Any compression achieved by employing **LZW** in your above encoding process?

***	4	3	7
***	4	3	7
***	4	3	7
***	4	3	7

Fig. A 3-bit image

Solution:
Given in Origin42, 1(b)

Malware37

1.

a) Suppose we have a grayscale image with most of the values of pixels being same. What can we use to compress the size of the image? **[1+5]**

In Fig 1.1, *** is representing the last three digits of your ID (e.g. for ID 160104001, *** will be 001).

***	40	60	80	100
***	40	60	80	100
***	40	60	80	100
***	40	60	80	100
***	40	60	80	100

Fig 1.1 A 7-bit input image

Show your step by step **LZW** encoding process (To generate your codebook, use **9** bits). Any compression achieved by employing **LZW** in your above encoding process?

Solution:
A grayscale image with most of the pixel values being the same is a spatial redundancy. This can be overcome by many compression techniques involving spatial redundancy, most notably LZW coding.

The image is as follows:

39	40	60	80	100
39	40	60	80	100
39	40	60	80	100
39	40	60	80	100
39	40	60	80	100

Generating a 9-bit codebook:

Location	Entry
0 … 255	0 … 255
256 … 511	- … -

Using LZW:

Current Recognized sequence	Pixel being processed	Encoded output	Dictionary Location (codeword)		Dictionary entry
	39				
39	40	39	256		39-40
40	60	40	257		40-60
60	80	60	258		60-80
80	100	80	259		80-100
100	39	100	260		100-39
39	40	-	-		-
39-40	60	256	261		39-40-60

60	80	-	-		-
60-80	100	258	262		60-80-100
100	39	-	-		-
100-39	40	260	263		100-39-40
40	60	-	-		-
40-60	80	257	264		40-60-80
80	100	-	-		-
80-100	39	259	265		80-100-39
39	40	-	-		-
39-40	60	-	-		-
39-40-60	80	261	266		39-40-60-80
80	100	-	-		-
80-100	39	-	-		-
80-100-39	40	265	267		80-100-39-40
40	60	-	-		-
40-60	80	-	-		-
40-60-80	100	264	268		40-60-80-100

Location	Entry
0 … 255	0 … 255
256	39-40
257	40-60
258	60-80
259	80-100
260	100-39

261	39-40-60
262	60-80-100
263	100-39-40
264	40-60-80
265	80-100-30
266	39-40-60-80
267	80-100-39-40
268	40-60-80-100
269 … 511	- … -

Before compression,
Total outputs = 25
Bit size = 8
So, image size = 25*8 = 200

After compression,
Total outputs = 13
Bit size = 9
So, image size = 13*9 = 117

Compression ratio = 200 : 117 = **1.709 : 1**
So, compression is **achieved**.

COREi36

3.

a) What is data redundancy? How many types of redundancies are there in an image? Give brief description of each. [4]

L-15

Solution:

b) Why do we need image compression? Consider the simple 4×8, 8-bit image: [6]

2	2	2	9	16	24	43	43
2	2	2	9	16	24	43	43
2	2	2	9	16	24	43	43
2	2	2	9	16	24	43	43

i. Compute the entropy of the image.
ii. Compress the image using Huffman coding.
iii. Compute the compression achieved and the effectiveness of the Huffman coding.

Solution:

c) Define the compression technique **Golomb** code of a nonnegative integer n with respect to a positive integer m denoted by $G_m(n)$. Compute Golomb code for $G_4(9)$. [4]

Solution: