



Image Compression

Lec-15

By

Dr. Omar F. Mohammad

Image Compression Fundamentals

Image Compression is reducing the size of the image data while retaining the necessary data for the image file. The shrinking file size is called the compressed file that is used to retrieve the compressed image.

هو تقليل حجم بيانات الصورة مع الاحتفاظ بالبيانات الضرورية لملف الصورة اي ان حجم الملف المتقلص يسمى بالملف المضغوط الذي يستخدم لاسترجاع الصورة المضغوطة.

We should know:

1. Image Type
2. Image Size
3. Image data

The compression process consists of two phases:

1. Image compression
2. Decompression

Image Compression Fundamentals

Data compression: is the process of reducing the amount of data required to represent given quantity of information. The **data** and **information** are not the same meaning.

Data **refers to the means by which the information is conveyed.** البيانات هي الوسائل التي يتم من خلالها نقل المعلومات.

Various amounts of **data** can represent the **same amount of information.**

Data redundancy: is the central concept in image compression and can be mathematically defined.

The **goal of image compression** is to reduce the amount of data required to represent a digital image.

Image compression methods:

1. Lossless (**Reversible** يمكن ارجاعها)
2. Lossy (**Non reversible** لا يمكن ارجاعها)

Lossless Data Compression

- In **lossless data compression**, the integrity of the data is preserved يحافظ عليها. The original data and the data after compression and decompression are exactly the same because; the **compression** and **decompression algorithms** are **exact inverses** of each other is no part of the data is lost in the process.
- **Redundant** المكرر data is **removed** in compression and **added** during decompression. Lossless compression methods are normally used when we cannot afford to lose any data. In general the lossless method :
 1. **Information preserving.** الحفاظ على المعلومات
 2. **Low compression ratios.**

Lossy (Non reversible)

- Our eyes and ears cannot distinguish subtle الطفيفة changes. In such cases, we can use a **lossy data** compression method. These methods **are cheaper**; they take **less time** and **space** when it comes to sending **millions of bits per second** for images and video.

Several methods have been developed using lossy compression techniques:

1. **JPEG (Joint Photographic Experts Group) encoding** is used to compress pictures and graphics
2. **MPEG (Moving Picture Experts Group) encoding** is used to compress video, and
3. **MP3 (MPEG audio layer 3)** for audio compression.

In general the lossy method:

1. **Not information preserving**
2. **High compression ratios**

Image Compression Fundamentals

Definitions

n_1 = data.

n_2 = data redundancy (i.e., data after compression).

Compression ratio (C_R) = $\frac{n_1}{n_2}$

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

Example:

if $C_R = 10$, then $R_D = \left(1 - \frac{1}{10}\right) \times 100 = (0.9) \times 100 = 90\%$

((90% of data in *dataset1* is redundant))

Image Compression Fundamentals

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

There are several ways of lossless method we can take some of them as follows:

1. Huffman Coding
2. Run-Length coding (RLC).
3. Difference coding (**predictive coding**)

Yields the smallest possible number of unique code symbols per source symbol.

يحقق أصغر عدد ممكن من الرموز الفريدة لكل مصدر رمز.

Basic idea: Different gray levels occur with **different probability** (non- uniform histogram).

Code: a list of symbols (**letters, numbers, bits** etc.)

Code word: a sequence of symbols used to represent a piece of information or an event (e.g., gray levels).

Code word length: number of symbols in each code word

Image Compression Fundamentals

Example: (binary code, symbols: 0,1, length: 3)

0: 000	4: 100
1: 001	5: 101
2: 010	6: 110
3: 011	7: 111

Image Compression Fundamentals

- Use shorter code words for the more common gray levels and longer code words for the less common gray levels. This is called Variable Length Coding.
- The amount of data in an $M \times N$ image with L gray levels is equal to $M \times N \times L_{\text{avg}}$, where:
-

$$\text{Average number of bits}(L_{\text{avg}}) = \sum_{k=0}^{L-1} l(r_k) P(r_k)$$

$l(r_k)$ is the number of bits used to represent gray level r_k , and $P(r_k)$ is the probability of gray level r_k in the image.

Total number of bits = $N * M * L_{\text{avg}}$

Huffman Coding

Huffman Coding steps:

Step 1.

1. Sort the gray levels by **decreasing** probability.
2. Add the **two smallest** probabilities.
3. Sort the **new value** into the list.
4. **Repeat** until only two probabilities remain.

Example :

$$a_1 = 0.1, a_2 = 0.4, a_3 = 0.06$$

$$, a_4 = 0.1, a_5 = 0.04, a_6 = 0.3$$

Original source		Source reductions			
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

Huffman Coding steps:

Step 2.

1. Give the **code 0** to the **highest probability** and the **code 1** to the **lowest probability** in the present node.
2. Go backwards through the tree and **add 0 to the highest** and **1 to the lowest** probability in each node **until** all gray levels have a unique code.

Original source			Source reductions							
Symbol	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

L_{avg} assuming Huffman coding:

$$\begin{aligned} L_{avg} &= \sum_{k=1}^6 l(a_k)P(a_k) \\ &= 3 \times 0.1 + 1 \times 0.4 + 5 \times 0.06 + 4 \times 0.1 + 5 \times 0.04 + 2 \times 0.3 \\ &= 2.2 \text{ bit/symbol} \end{aligned}$$

L_{avg} assuming binary codes: 6 symbol, we need a 3-bit code

$a_1: 000, a_2: 001, a_3: 010, a_4: 011, a_5: 100, a_6: 101$

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

Huffman Coding

Example

r_k	$p(r_k)$	node 1	node 2	node 3	node 4	node 5	node 6
1	0.4	→ 0.4	0.4	0.4	0.4	0.4	↘ 0.6
4	0.3	→ 0.3	0.3	0.3	0.3	0.3	↗ 0.4
0	0.1	→ 0.1	0.1	0.1	↘ 0.2	→ 0.3	
5	0.1	→ 0.1	0.1	0.1	↗ 0.1	↗	
3	0.05	→ 0.05	0.05	→ 0.1			
2	0.03	→ 0.03	→ 0.05	↗			
6	0.01	→ 0.02	↗				
7	0.01	↗					
$L_{avg} = 3$							
r_k	code	node 1	node 2	node 3	node 4	node 5	node 6
1	1	1	1	1	1	1	↙ 0
4	00	00	00	00	00	00	↖ 1
0	011	011	011	011	↙ 010	← 01	
5	0100	0100	0100	0100	↖ 011	↖	
3	01010	01010	01010	← 0101			
2	010110	010110	← 01011	↖			
6	0101110	← 010111	↖				
7	0101111	↖					

Huffman Coding

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) = 2.27$$

$$C_R = \frac{n_1}{n_2} = \frac{3}{2.27} = 1.32$$

$$R_D = 1 - \frac{1}{C_R} = \frac{n_1 - n_2}{n_1} = \frac{3 - 2.27}{3} = 0.24$$

1. The Huffman code results in **unambiguous code**.
2. The code is **reversible without loss**.
3. The table for **the translation of the code** has to be **stored together** with the coded image.
4. The Huffman code does not **take correlation** between **adjacent pixels** into consideration

Run-Length Coding (RLC)

2. Run-Length coding (RLC).

Every code word is made up of a **pair** (**g**, **l**) where **g** is the gray level, and **l** is the number of pixels with that gray level (length, or “run”).

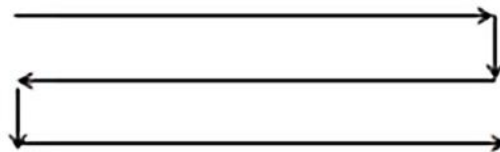
Example:

56 56 56 82 82 82 83 80

56 56 56 56 56 80 80 80

Creates the run-length code (56, 3)(82, 3)(83, 1)(80, 4)(56, 5).

The code is calculated row by row.



Very efficient coding for binary data. Important to know position, and the image dimensions must be stored with the coded image. Used in most fax machines.

Difference coding (Predictive Coding)

3. Difference coding

$$f(x) = \begin{cases} x_i & \text{if } i = 0 \\ x_i - x_{i-1} & \text{if } i > 0 \end{cases}$$

Example

Original = 56 56 56 82 82 82 83 80 80 80 80 56 56 56 56 56

Code $f(x_i)$ = 56 0 0 26 0 0 1 -3 0 0 0 -24 0 0 0 0

Both **run-length coding**, and **difference coding** are reversible, and can be combined with, e.g., **Huffman coding**.

Image Compression Fundamentals

Example: Consider the pixel {23, 34, 39, 47, 55, 63}. Demonstrate the predictive coding.

Solution:

Value	Predictive Coding
23	23
34	$34 - 23 = 11$
39	$39 - 34 = 5$
47	$47 - 39 = 8$
55	$55 - 47 = 8$
63	$63 - 55 = 8$

- **Max value in original sequence = 63** **Requires (6-bits + 1-bit for sign)**
- **No. of bits required to code original message = $6*7 = 42$**
- **Max value in predicted code is = 23** **Requires (5-bits + 1-bit for sign)**
- **No. of bits required to code predictive code = $6*6 = 36$**


Image Compression Fundamentals

- ❑ If the **difference** crosses the **threshold limit**, it creates a problem known as **Overloading**
- ❑ Solution of this is to ignore the differences and use the original message for coding

Consider the pixel {23, **64**, 39, 47, 55, 63}

Value	Predictive Coding
23	23
64	$64 - 23 = 41$
39	$39 - 64 = -25$
47	$47 - 39 = 8$
55	$55 - 47 = 8$
63	$63 - 55 = 8$

- **Max value in original sequence = 63** **Requires (6-bits + 1-bit for sign)**
- **No. of bits required to code original message = $6 * 7 = 42$**
- **Solution**
- $5 * (5+1) + 1(6+1) \implies 5*6+7$
- $30 + 7 = 37$



End of Lecture