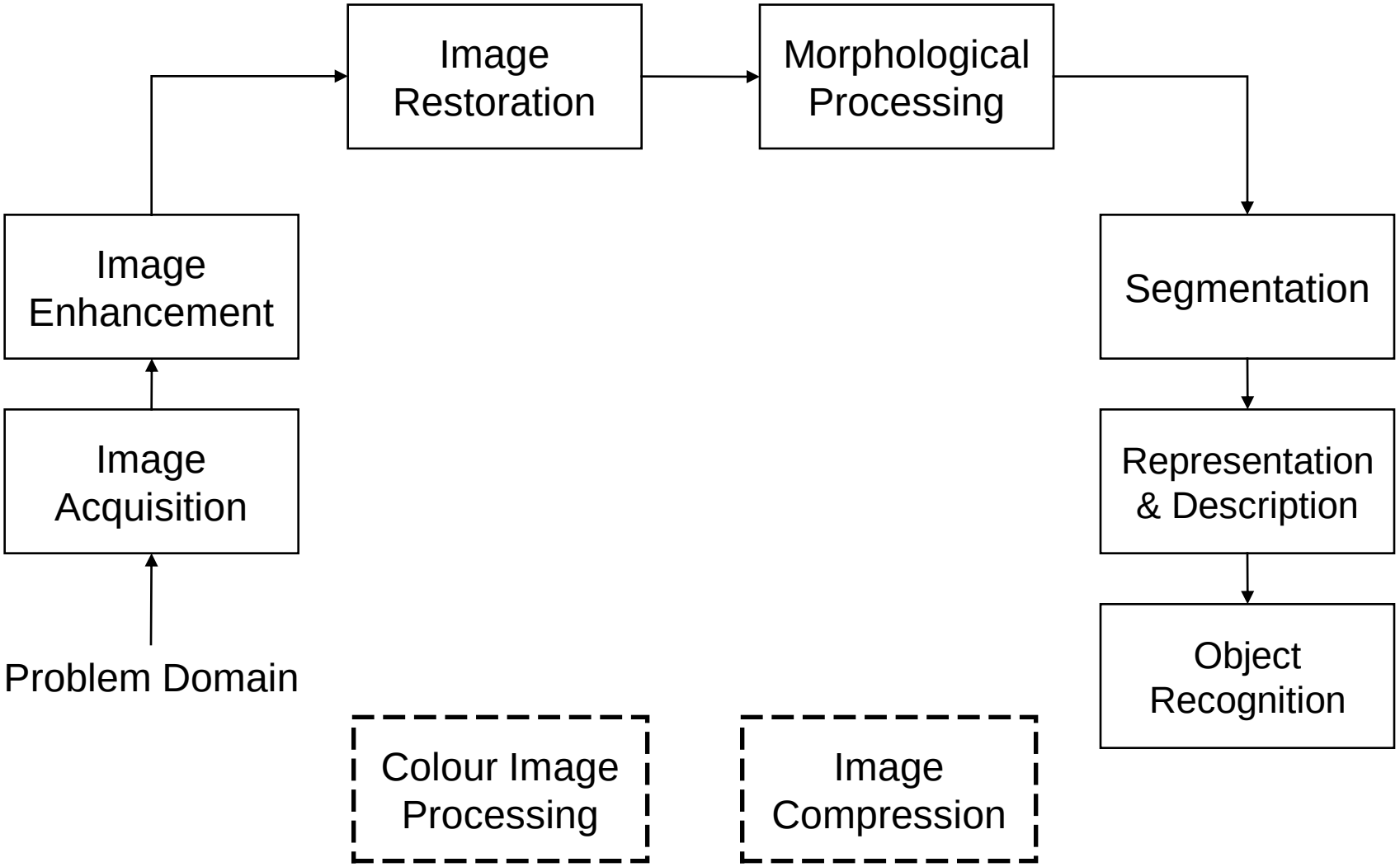
Image Compression-I

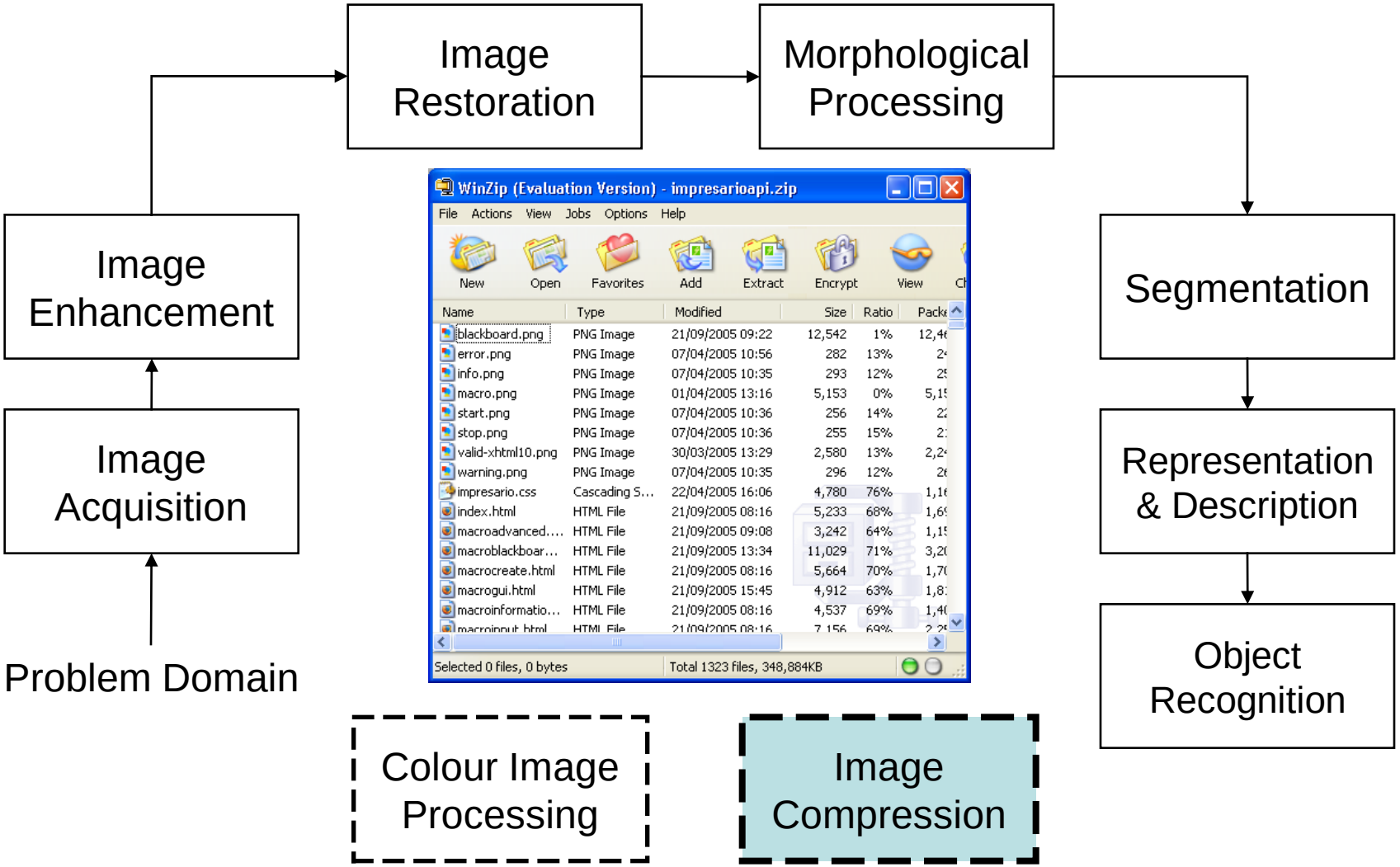
Contents

- ✓ Overview
- ✓ Lossless Compression Techniques
  - Run Length Coding

Phases of Digital Image Processing



Phases of Digital Image Processing : Image Compression



Introduction and Overview

The field of image compression continues to grow at a rapid pace.

In future, the need to store and transmit images will only continue to increase faster than the available capability to process all the data.

Purpose of Compression

- Gray Scale 8-bit Image: 720×480 = 345.6 KB
- Color Image: 720×480 = 1.04 MB
- 90 minutes Color Video: 720×480×30frames/sec=168.48 GB
- Internet Speed: 56kbps
- Approx. download time: 51 seconds, 156 seconds and 299 days, respectively.
- Outcomes of Compression:
  - ✓ Reduce Storage Space
  - ✓ Reduce Transmission Time
  - ✓ Reduce Data access Time

Different Applications of Image Compression

- Internet
- Satellite imaging
- Medical imaging
- Multimedia
- Businesses

Compression algorithm development starts with applications to two-dimensional (2-D) still images.

After the 2-D methods are developed, they are often extended to video (motion imaging).

We will discuss image compression of single frames of image data.

Image compression involves reducing the size of image data files, while retaining necessary information.

Retaining necessary information depends upon the application.

Image segmentation methods, which are primarily a data reduction process, can be used for compression.

The reduced file created by the compression process is called the compressed file and is used to reconstruct the image, resulting in the decompressed image.

The original image, before any compression is performed, is called the uncompressed image file.

The ratio of the uncompressed image file and the compressed file is referred to as the Compression Ratio.

Compression Ratio

$$Compression\ Ratio = \frac{Uncompressed\ file\ size}{Compressed\ file\ size} = \frac{SIZE_U}{SIZE_C}$$

$$Often\ writen\ as\ \rightarrow\ SIZE_U : SIZE_C$$

EXAMPLE 10.1.1: The original image is 256x256 pixels, single-band (grayscale), 8-bits per pixel. This file is 65,536 bytes (64k). After compression the image file is 6,554 bytes. The compression ratio is: $SIZE_U/SIZE_C = 65536/6554 = 9.999 \approx 10$. This can also be written as 10:1

This is called a "10 to 1 compression", a "10 times compression", or can be stated as

"compressing the image to 1/10 its original size". Another way to state the compression is to use

the terminology of *bits per pixel*. For an NxN image:

$$Bits\ per\ pixel = \frac{Number\ of\ bits}{Number\ of\ pixels} = \frac{(8)(Number\ of\ bytes)}{NxN}$$

EXAMPLE 10.1.2: Using the preceding example, with a compression ratio of 65,536/6,554 bytes, we want to express this as bits per pixel. This is done by first finding the number of pixels in the image: 256x256 = 65,536 pixels. We then find the number of bits in the compressed image file: (6,554 bytes)(8 bits/byte) = 52,432 bits. Now we can find the bits per pixel by taking the ratio: 52,432/65,536 = 0.8 bits/pixel

The reduction in file size is necessary to meet the bandwidth requirements for many transmission systems, and for the storage requirements in computer databases.

To understand "*retaining necessary information*", we must differentiate between data and information.

1. Data:

- For digital images, data refers to the pixel gray level values that correspond to the brightness of a pixel at a point in space.

- Data are used to convey information, much like the way the alphabet is used to convey information via words.

2. Information:

- Information is an interpretation of the data in a meaningful way.

- Information can be application specific.

Types of Image Compression Methods

1. Lossless Compression Methods:

- Allows for the exact restoration of the original image data.

- It can compress complex images to a maximum 1/2 to 1/3 the original size – 2:1 to 3:1 compression ratios.

- Preserves the data exactly.

2. Lossy Compression Methods:

- Data loss, original image cannot be re-created exactly.

- It can compress complex images 10:1 to 50:1 and retain high quality, and 100 to 200 times for lower quality, but acceptable images.

Compression algorithms are developed by taking advantage of the redundancy that is inherent in image data.

Four primary types of redundancy that can be found in images are:

1. Coding
2. Interpixel
3. Interband / Chromatic
4. Psychovisual redundancy

1. Coding Redundancy

It occurs when the data used to represent the image is not utilized in an optimal manner.

2. Interpixel Redundancy

It occurs because adjacent pixels tend to be highly correlated, in most images the brightness levels do not change rapidly, but change gradually.

3. Interband /Chromatic Redundancy

Occurs in color images due to the correlation between bands within an image – if we extract the red, green, and blue bands they look similar.

4. Psychovisual Redundancy

Some information is more important to the human visual system than other types of information.

The key in image compression algorithm development is to determine the minimal data required to retain the necessary information.

The compression is achieved by taking advantage of the redundancy that exists in images.

If the redundancies are removed prior to compression, for example with a decorrelation process, a more effective compression can be achieved.

Compression System Model

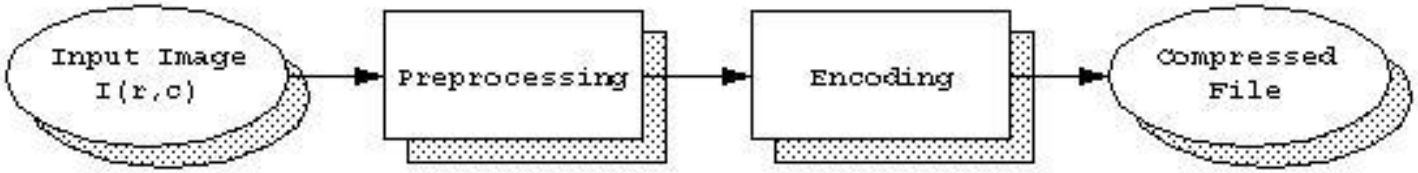
It consists of two parts:

1. The compressor
2. The decompressor

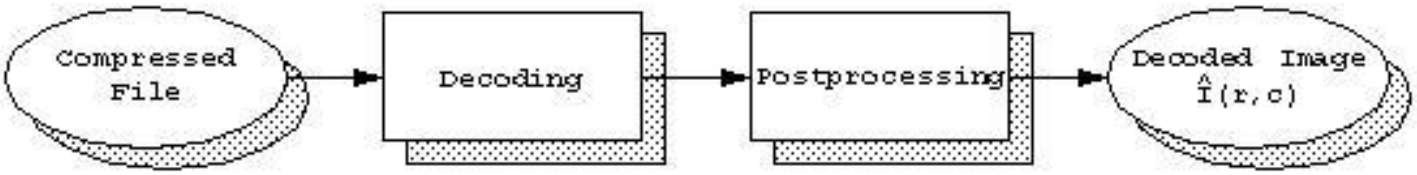
The compressor consists of a preprocessing stage and encoding stage.

The decompressor consists of a decoding stage followed by a postprocessing stage.

Figure 10.1-1: Compression System Model



a) Compression

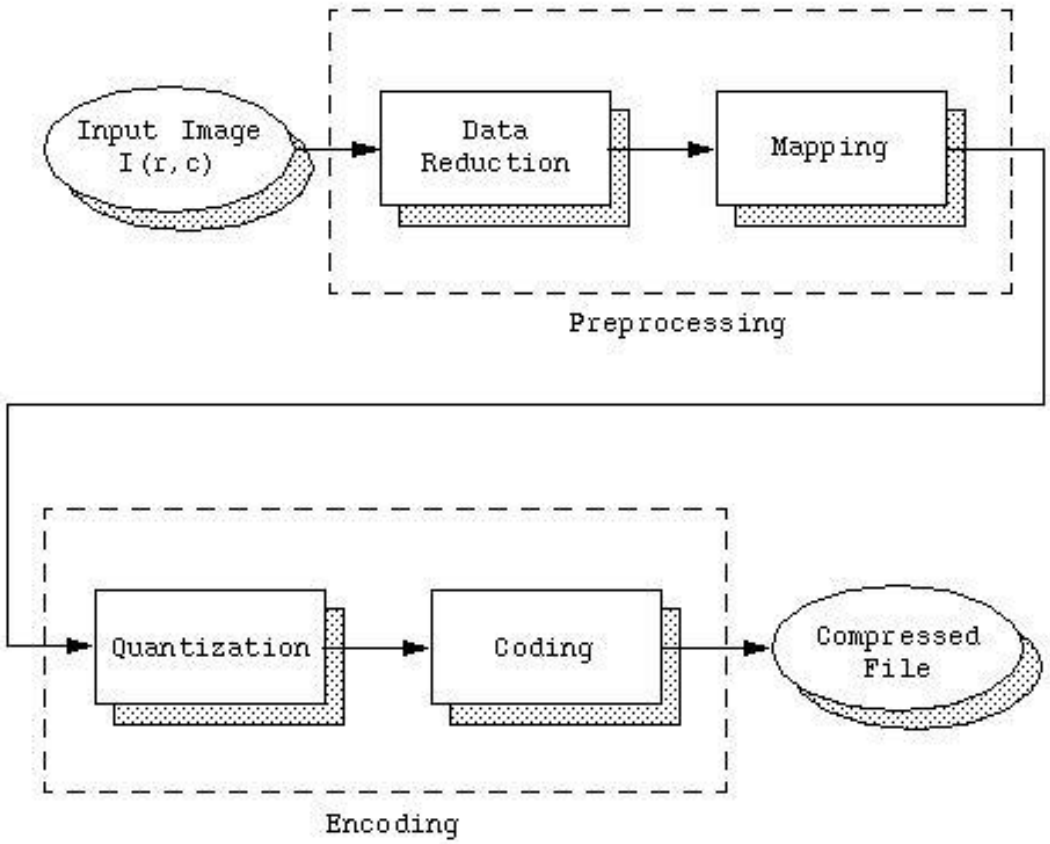


b) Decompression

Before encoding, preprocessing is performed to prepare the image for the encoding process, and consists of any number of operations that are application specific.

After the compressed file has been decoded, postprocessing can be performed to eliminate some of the potentially undesirable artifacts brought about by the compression process.

Figure 10.1-2: The Compressor



Stages of Compressor

1. Data Reduction: Image data can be reduced by gray level and/or spatial quantization, or can undergo any desired image improvement (for example, noise removal) process.

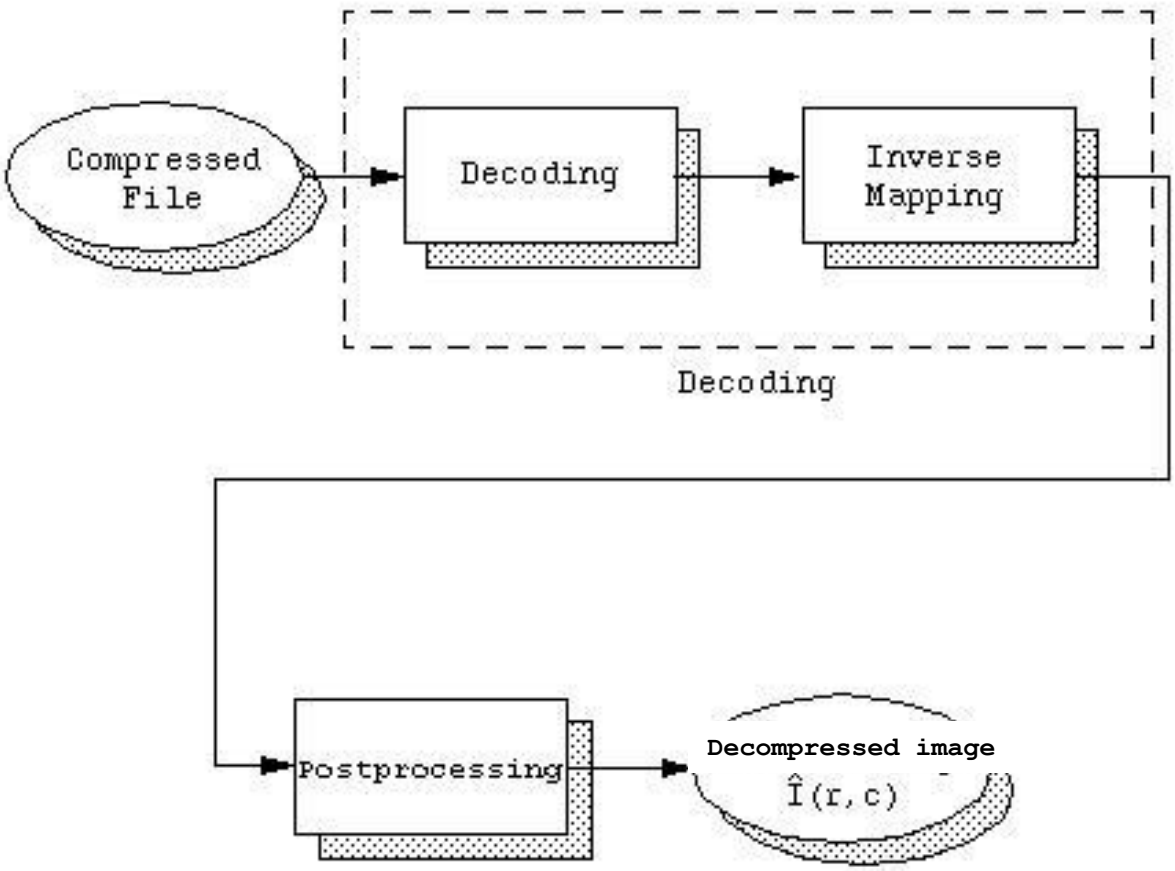
2. Mapping: Involves mapping the original image data into another mathematical space where it is easier to compress the data.

3. Quantization: Involves taking potentially continuous data from the mapping stage and putting it in discrete form.

4. Coding: Involves mapping the discrete data from the quantizer onto a code in an optimal manner.

A compression algorithm may consist of all the stages, or it may consist of only one or two of the stages.

Figure 10.1-3: The Decompressor



Stages of Decompressor

1. Decoding: Takes the compressed file and reverses the original coding by mapping the codes to the original, quantized values.

2. Inverse mapping: Involves reversing the original mapping process.

3. Postprocessing: Involves enhancing the look of the final image.

This may be done to reverse any preprocessing, for example, enlarging an image that was shrunk in the data reduction process.

In other cases, postprocessing may be used to simply enhance the image to improve any artifacts from the compression process itself.

The development of a compression algorithm is highly application specific.

Preprocessing stage of compression consists of processes such as enhancement, noise removal, or quantization.

The goal of preprocessing is to prepare the image for the encoding process by eliminating any irrelevant information, where irrelevant is defined by the application.

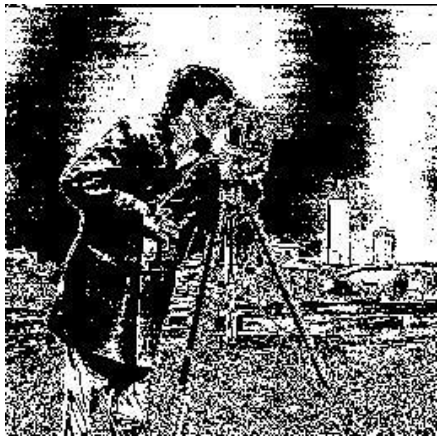
For example, many images that are for viewing purposes only can be preprocessed by eliminating the lower bit planes, without losing any useful information.

a) Original image

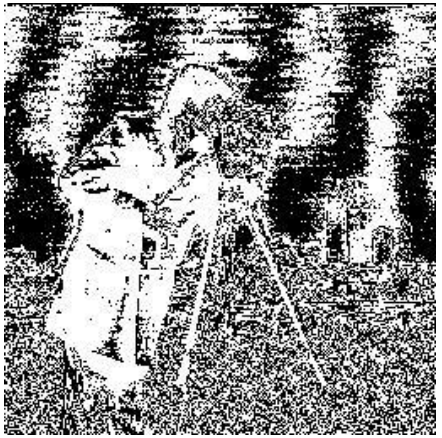
b) Bit plane 7, the most significant bit

c) Bit plane 6

d) Bit plane 5

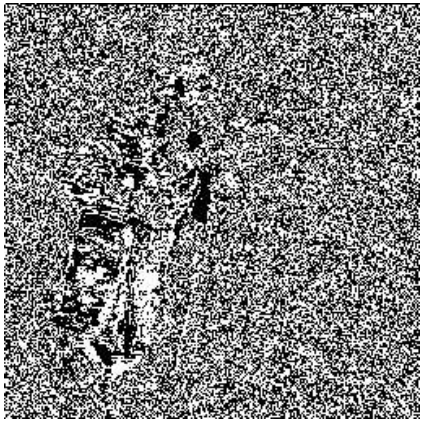


e) Bit plane 4

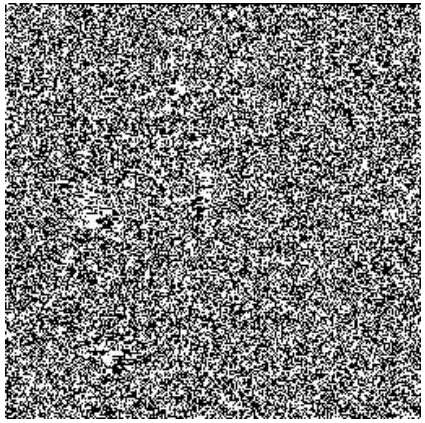


f) Bit plane 3

g) Bit plane 2



h) Bit plane 1



i) Bit plane 0,
the least significant bit

Lossless Compression Methods

No loss of data, decompressed image exactly same as uncompressed image.

Medical images or any images used in benches.

Lossless compression methods typically provide about a 10% reduction in file size for complex images.

The underlying theory for lossless compression (also called *data compaction*) comes from the area of communications and information theory, with a mathematical basis in probability theory.

One of the most important concepts used is the idea of information content and randomness in data.

Entropy is the measurement of the average information content in an image.

The entropy for an $N \times N$ image is calculated by this equation:

$$Entropy = -\sum_{i=0}^{L-1} p_i \log_2 p_i \text{ (in bits/pixel)}$$

Where,

p_i= the probability of the $i^{th} gray\ level = \frac{n_i}{N^2}$

n_i= the total number of pixels with gray value i.

L= the total number of gray levels.

This measure provides us with a theoretical minimum for the average number of bits per pixel that could be used to code the image.

It can also be used as a metric for judging the success of a coding scheme, as it is theoretically optimal.

EXAMPLE 10.2.1:

Let L = 8, meaning there are 3 bits/pixel in the original image. Now, let's say the number of pixels at each gray level value is equal (they have the same probability), that is:

$$p_0 = p_1 = \ldots = p_7 = \frac{1}{8}$$

Now, we can calculate the entropy as follows:

$$Entropy = -\sum_{i=0}^{7} p_i \log_2(p_i) = -\sum_{i=0}^{7} \frac{1}{8} \log_2\left(\frac{1}{8}\right) = 3$$

This tells us that the theoretical minimum for lossless coding for this image is 3 bits per pixel. In other words, there is no code that will provide better results than the one currently used (called the natural code, since $000_2 = 0, 001_2 = 1, 010_2 = 2, \ldots, 111_2 = 7$). This example illustrates that the image with the most random distribution of gray levels, a uniform distribution, has the highest entropy

The range of the entropy:

$$0 \leq Entropy \leq \log_2(L)$$

The examples also illustrate the information theory perspective regarding information and randomness.

The more randomness that exists in an image, the more evenly distributed the gray levels, and more bits per pixel are required to represent the data.

a) Original image,
entropy = 7.032 *bpp*

b) Image after local histogram equalization,
block size 4, entropy = 4.348 *bpp*

c) Image after binary threshold,
entropy = 0.976 *bpp*

Run-Length Coding

Run-Length Coding

Run-length coding (RLC) works by counting adjacent pixels with the same gray level value called the *run-length,* which is then encoded and stored.

RLC works best for binary, two-valued, images.

RLC can also work with complex images that have been preprocessed by thresholding to reduce the number of gray levels to two.

RLC can be implemented in various ways, but the first step is to define the required parameters.

Horizontal RLC (counting along the rows) or vertical RLC (counting along the columns) can be used.

In basic horizontal RLC, the number of bits used for the encoding depends on the number of pixels in a row.

If the row has 2^n pixels, then the required number of bits is *n*, so that a run that is the length of the entire row can be encoded.

EXAMPLE 10.2.5:

A 256x256 image requires 8-bits, since $2^8 = 256$.

EXAMPLE 10.2.6:

A 512x512 image requires 9-bits, since $2^9 = 512$.

The next step is to define a convention for the first RLC number in a row – does it represent a run of 0's or 1's?

EXAMPLE 10.2.7:

The image is an 8x8 binary image, which requires 3 bits for each run-length coded word. In the actual image file are stored 1's and 0's, although upon display the 1's become 255 (white) and the 0's are 0 (black). To apply RLC to this image, using horizontal RLC:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

EXAMPLE 10.2.7 (contd):

The RLC numbers are:

First row: 8

Second row: 0, 4, 4

Third row: 1, 2, 5

Fourth row: 1, 5, 2

Fifth row: 1, 3, 2, 1, 1

Sixth row: 2, 1, 2, 2, 1

Seventh row: 0, 4, 1, 1, 2

Eighth row: 8

Note that in the second and seventh rows, the first RLC number is 0, since we are using the convention that the first number corresponds to the number of zeros in a run

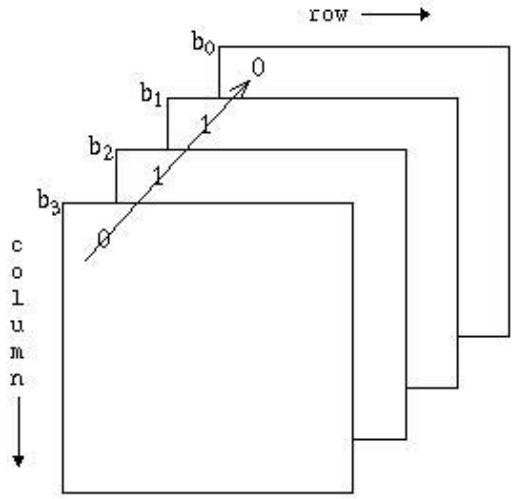
Bitplane-RLC : A technique which involves extension of basic RLC method to gray level images, by applying basic RLC to each bit-plane independently.

For each binary digit in the gray level value, an image plane is created, and this image plane (a string of 0's and 1's) is then encoded using RLC.

Figure 10.2-4: Bit Plane Run-length Coding

b_3	b_2	b_1	b_0
0	0	0	0
0	0	0	1
0	0	1	0
.	.	.	.
.	.	.	.
.	.	.	.
0	1	1	0
1	1	1	1

a) 4 bits/pixel designation



b) bit-planes → b3 b2 b1 b0
0 1 1 0

Another way to extend basic RLC to gray level images is to include the gray level of a particular run as part of the code.

Here, instead of a single value for a run, two parameters are used to characterize the run.

The pair (G,L) correspond to the gray level value G, and the run length L.

This technique is only effective with images containing a small number of gray levels.

EXAMPLE 10.2.9:

Given the following 8x8, 4-bit image:

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 12 & 12 & 12 \\ 10 & 10 & 10 & 10 & 10 & 12 & 12 & 12 \\ 0 & 0 & 0 & 10 & 10 & 10 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 10 & 10 & 9 & 9 & 10 \\ 5 & 5 & 5 & 4 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

EXAMPLE 10.2.9 (contd):

The corresponding gray levels pairs are as follows:

First row: 10,8

Second row: 10,5 12,3

Third row: 10,5 12,3

Fourth row: 0,3 10,3 0,2

Fifth row: 5,3 0,5

Sixth row: 5,3 10,2 9,2 10,1

Seventh row: 5,3 4,3 0,2

Eighth row: 0,8

These numbers are then stored in the RLC compressed file as:

10,8,10,5,12,3,10,5,12,3,0,3,10,3,0,2,5,3,0,5,5,3,10,2,9,2,10,1,5,3,4,3,0,2,0,8

The decompression process requires the number of pixels in a row, and the type of encoding used.

Standards for RLC have been defined by the International Telecommunications Union-Radio (ITU-R, previously CCIR).

These standards use horizontal RLC, but postprocess the resulting RLC with a Huffman encoding scheme.

Newer versions of this standard also utilize a two-dimensional technique where the current line is encoded based on a previous line, which helps to reduce the file size.

These encoding methods provide compression ratios of about 15 to 20 for typical documents.