Run-length Coding

Method that works by counting the number of adjacent pixels with the same grey-level value. This count, called the run length, is then coded and stored.

Basic methods are used primarily for binary images, but can be used for more complex images that have been pre-processed by thresholding to reduce the number of gray levels to two.

Run-length Coding

Basic concept is to code each contiguous group of 1's or 0's encountered in a left to right scan of a row by its length, and establishing a convention for determining the length of the run.

The most common approaches for determining the value of the run are:-

(i) specify the value of the first run of each row, or

(ii) assume that the row begins with a white run, whose run-length may in fact be zero.

Basic RLC

Step1: - Define the required parameters.

horizontal or vertical RLC ?

Horizontal RLC – no.of bits for the coding depends on the no.of pixels in a row.

i.e. if row has 2^n pixels, required no.of bits = n

e.g.1. 256 x 256 image requires 8 bits, since $2^8 = 256$

e.g.2. 512 x 512 " " 9 " " $2^9 = 512$

Basic RLC

Step 2:- Define a convention for the first RLC number in a row – does it represent a run of 1's or 0's.

e.g. 8 x 8 binary image, which requires 3 bits for each run-length coded word

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

using horizontal RLC we get:

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
eigth row: 8

RLC compressed file

8, 0, 4, 4, 1, 2, 5, 1, 5, 2, 1, 3, 2, 1, 1, 2, 1, 2, 2, 1, 0, 4, 1, 1, 2, 8

Extended RLC

Used for gray level images.

Technique used called bit-plane RLC.

Decomposes multi-level (monochrome or colour) image into a series of binary images and compresses each binary image via one of several well-known compression methods.

Typical compression ratios of 0.5 to 1.2 are achieved with complex 8-bit monochrome images; so without further processing, this is not a good compression technique for complex images.

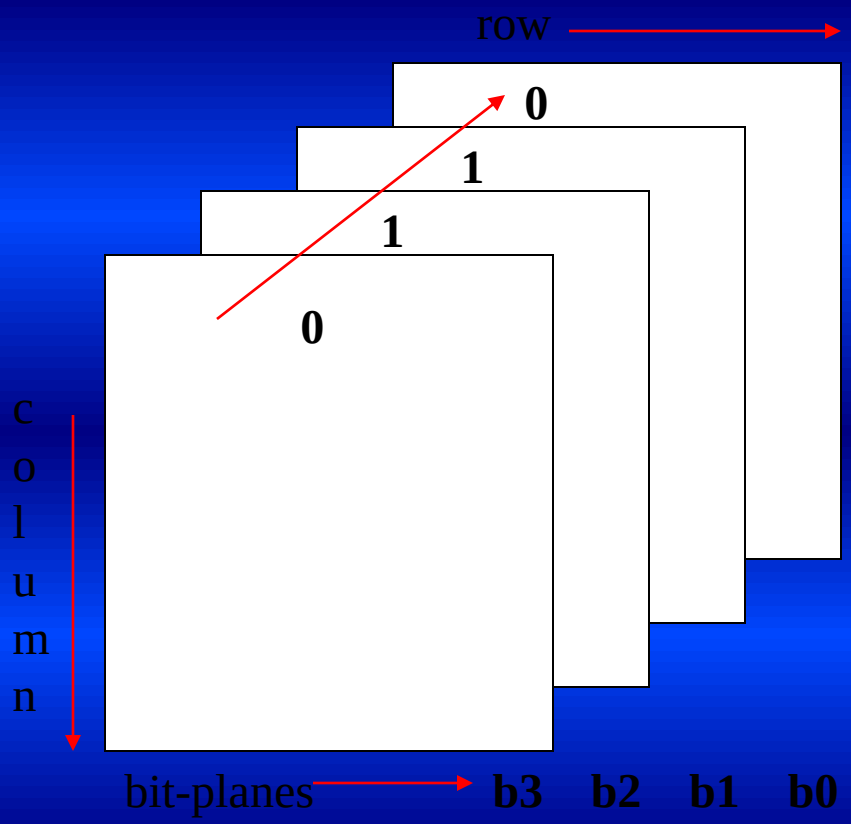
Bit plane RLC is most useful for simple images, such as graphics files, where much higher compression ratios are achieved.

Extended RLC

Step 1:- for each binary digit in the gray-level value a bit-plane is created.

b3	b2	b1	b0
0	0	0	0
0	0	0	1
0	0	1	0
.	.	.	.
.	.	.	.
0	1	1	0
1	1	1	1

4 bits/pixel designation



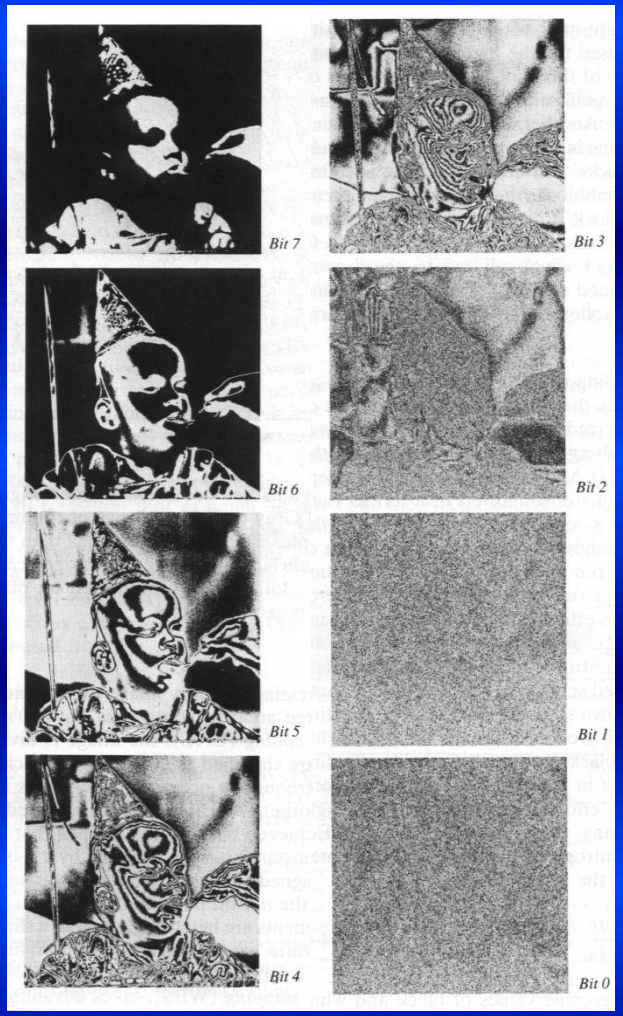
Extended RLC

Step 2:- code image plane (a string of 1's and 0's) using RLC

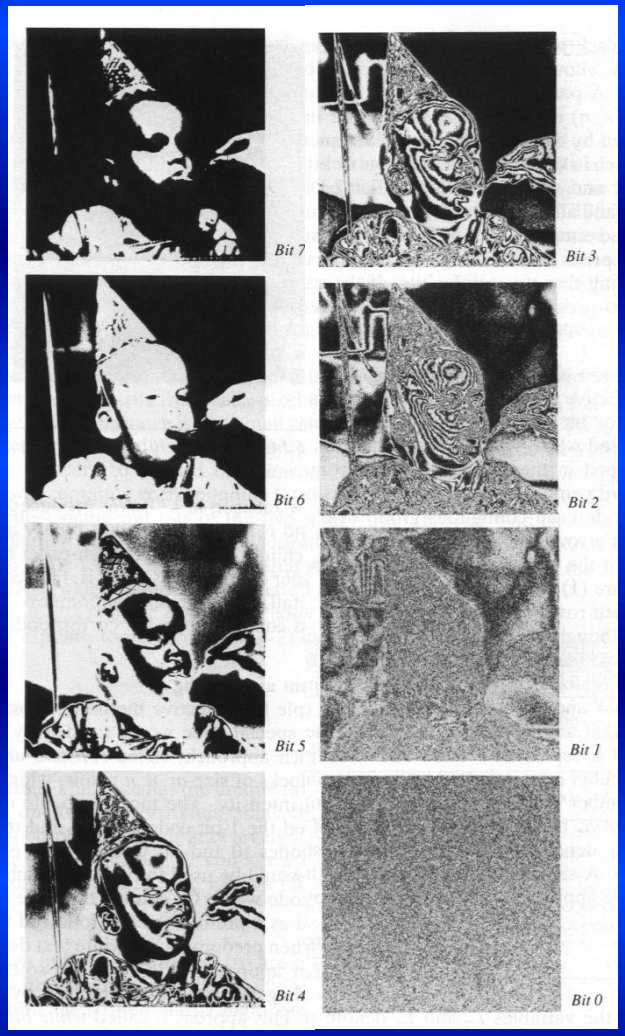
Decimal	4-bit Natural code	4-bit Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Natural code	Gray code
0111	0100
↓↓↓↓	↓↓↓↓
1000	1100

Bit-plane Coding - Example



Binary coded



Gray coded

Gray-level RLC

Instead of a single value for a run, 2 parameters are used to characterise the run.

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

Example 1: 8 x 8 4-bit image

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

first row: 10,8
second row: 10,5 12,3
third row: 10,5 12,3
fourth row: 0,3 10,3 0,3
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

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

RLC - Standards

Defined by the International Telecommunications Union-Radio (ITU-R)

Use horizontal RLC but postprocess the resulting RLC with a Huffmann encoding scheme.

Newer versions of this standard also use a two-dimensional technique where the current line is coded based on the previous line. This additional processing helps to reduce the file size.

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

Arithmetic Coding

In Arithmetic Coding there is no direct correspondence between the code and the individual pixel values.

Instead, an entire sequence of source symbols (or message) is assigned to a single arithmetic code.

It transforms input data into a single floating point number between 0 and 1

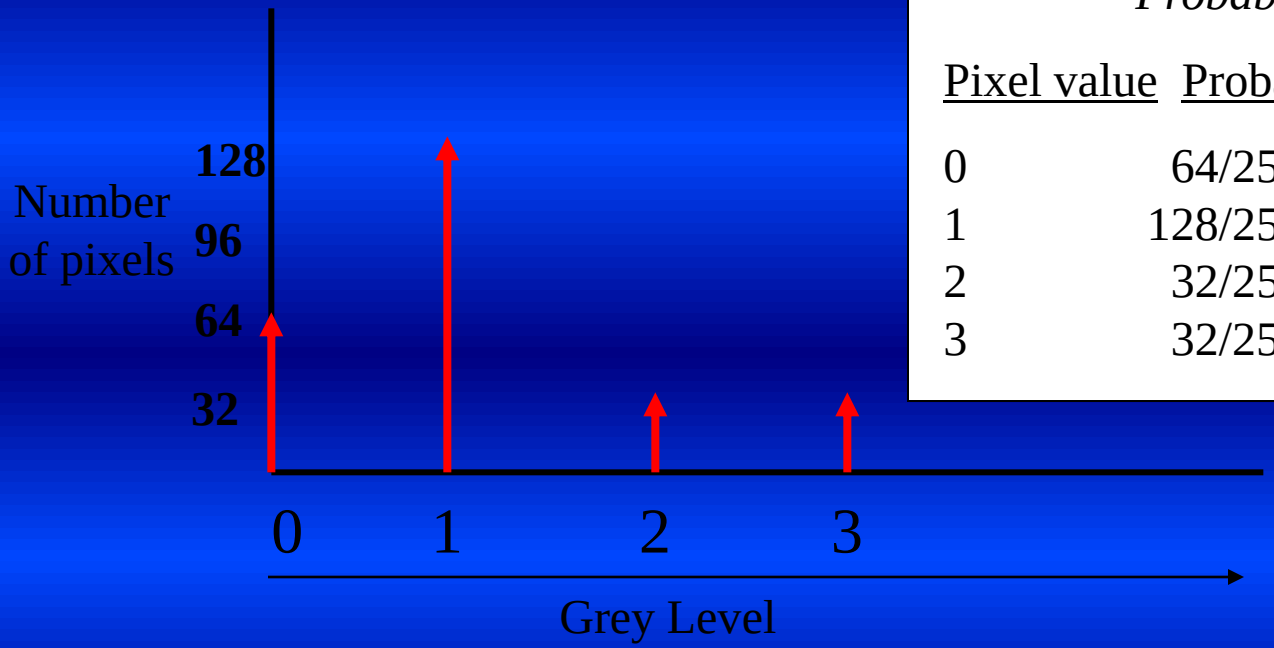
Arithmetic Coding

Entire image must be divided into small subimages to be encoded.

Arithmetic coding uses the probability distribution of the data (histogram), so it can theoretically achieve the maximum compression specified by the entropy.

Arithmetic Coding - Example

16 x 16 2 bit image – Histogram:



Probability Table:

Pixel value	Probability	Initial subinterval
0	64/256 = ¼	0 – ¼
1	128/256 = ½	¼ – ¾
2	32/256 = 1/8	¾ – 7/8
3	32/256 = 1/8	7/8 – 1

Arithmetic Coding - Example

1. Starting on the left, the initial 0 to 1 interval is subdivided, based on the probability distribution.

2. The first pixel value 0 is coded by extracting the subinterval corresponding to the 0 and subdividing it again, based on the same relative distribution.

3. Repeat process for each pixel value in the sequence until a final interval is determined, in this case from 58/1024 to 62/1024, or 0.056640625 to 0.0600546875. Any value within this subinterval, such as 0.057 or 0.060, can be used to represent this sequence of gray level values.

Arithmetic Coding - Example

Coding process illustrated with an example pixel value sequence of 0, 0, 3, 1