IMAGE COMPRESSION- I

Week VIII

Feb 25

Reading..

- Chapter 8
  - Sections 8.1, 8.2
  - 8.3 (selected topics)
  - 8.4 (Huffman, run-length, loss-less predictive)
  - 8.5 (lossy predictive, transform coding basics)
  - 8.6 Image Compression Standards (time permitting)

Image compression

Objective: To reduce the amount of data required to represent an image.

Important in data storage and transmission

- Progressive transmission of images (internet, www)
- Video coding (HDTV, Teleconferencing)
- Digital libraries and image databases
  - Medical imaging
  - Satellite images

IMAGE COMPRESSION

- Data redundancy
- Self-information and Entropy
- Error-free and lossy compression
- Huffman coding
- Predictive coding
- Transform coding

Lossy vs Lossless Compression

Compression techniques

Information preserving

(loss-less)

Images can be compressed and restored without any loss of information.
Application: Medical images, GIS

Lossy

Perfect recovery is not possible but provides a large data compression.
Example : TV signals, teleconferencing

Data Redundancy

- **CODING**: Fewer bits to represent frequent symbols.

- **INTERPIXEL / INTERFRAME**: Neighboring pixels have similar values.

- **PSYCHOVISUAL**: Human visual system can not simultaneously distinguish all colors.

Coding Redundancy

Fewer number of bits to represent frequently occurring symbols.

Let $p_r(r_k) = n_k / n$, k = 0,1,2, . ., L-1; L # of gray levels.

Let r_k be represented by $l(r_k)$ bits. Therefore average # of bits required to represent each pixel is

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

Usually $l(r_k) = m$ bits (constant). $\Rightarrow L_{avg} = \sum_k m\, p_r(r_k) = m$

Coding Redundancy (contd.)

- Consider equation (A): It makes sense to assign fewer bits to those r_k for which $p_r(r_k)$ are large in order to reduce the sum.
- this achieves data compression and results in a variable length code.
- More probable gray levels will have fewer # of bits.

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

Coding: Example

Example (From text)

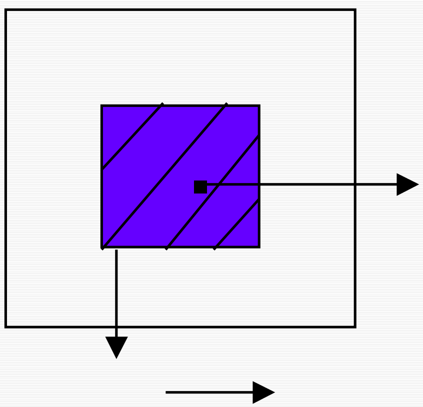
r_k	$p_r(r_k)$	Code	$l(r_k)$
$r_0 = 0$	0.19	11	2
$r_1 = 1/7$	0.25	01	2
$r_2 = 2/7$	0.21	10	2
$r_3 = 3/7$	0.16	001	3
$r_4 = 4/7$	0.08	0001	4
$r_5 = 5/7$	0.06	00001	5
$r_6 = 6/7$	0.03	000001	6
$r_7 = 1$	0.02	000000	6

$$L_{avg} = \sum \rho(r_k)\, l(r_k) = 2.7 \text{ Bits}$$

10% less code

Inter-pixel/Inter-frame

spatial

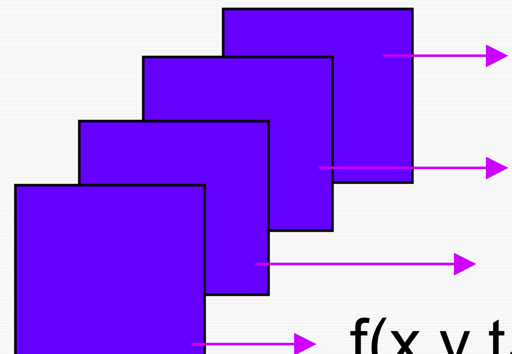


f(x,y) Depends on

f (x', y') , (x', y') ε N_{xy}

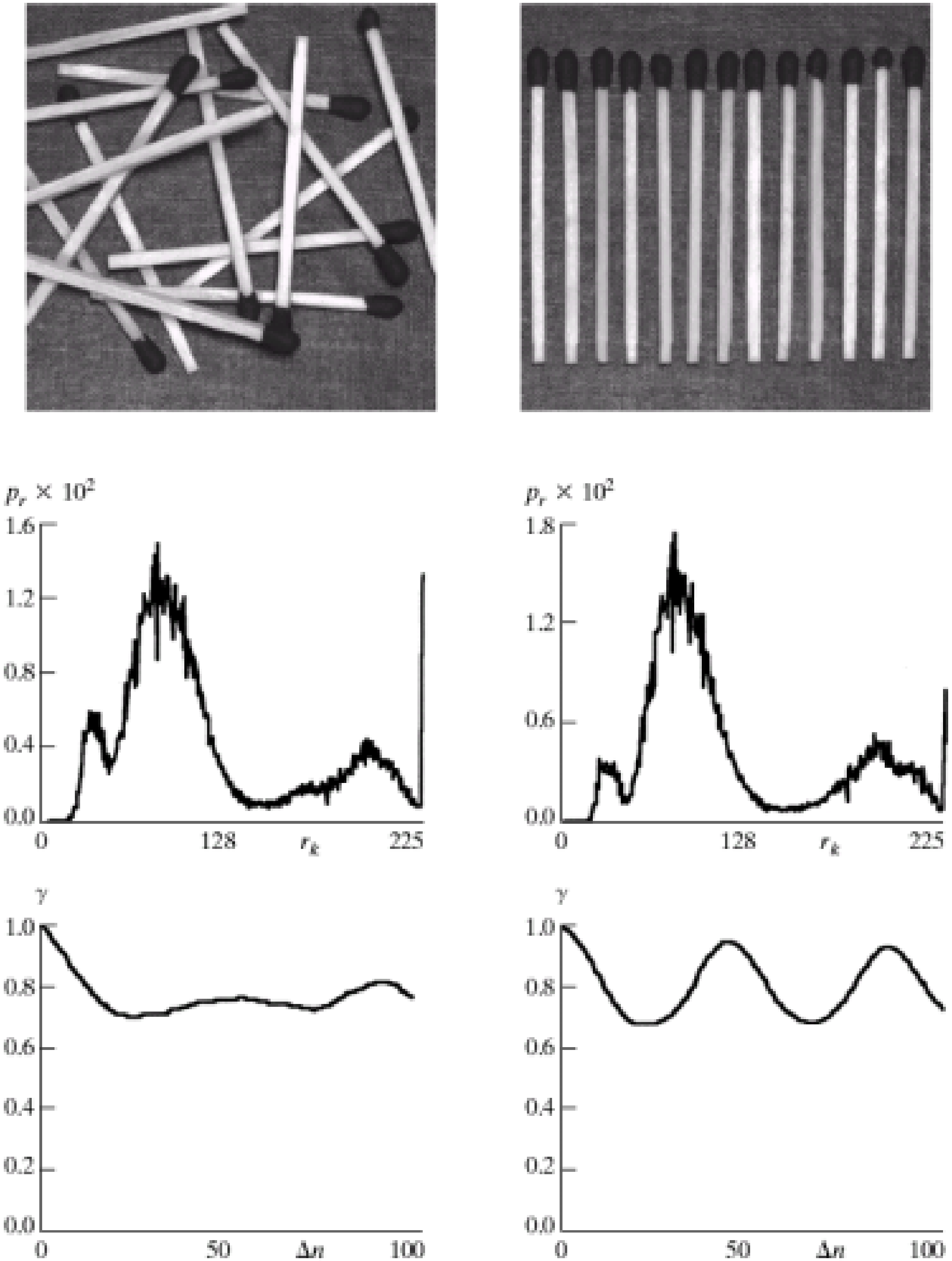
N_{xy} : Neighbourhood of pixels around (x,y)

Interframe



f(x, y, t_i) i=1, 2, 3, . . . are related to each other. This can be exploited for video compression.

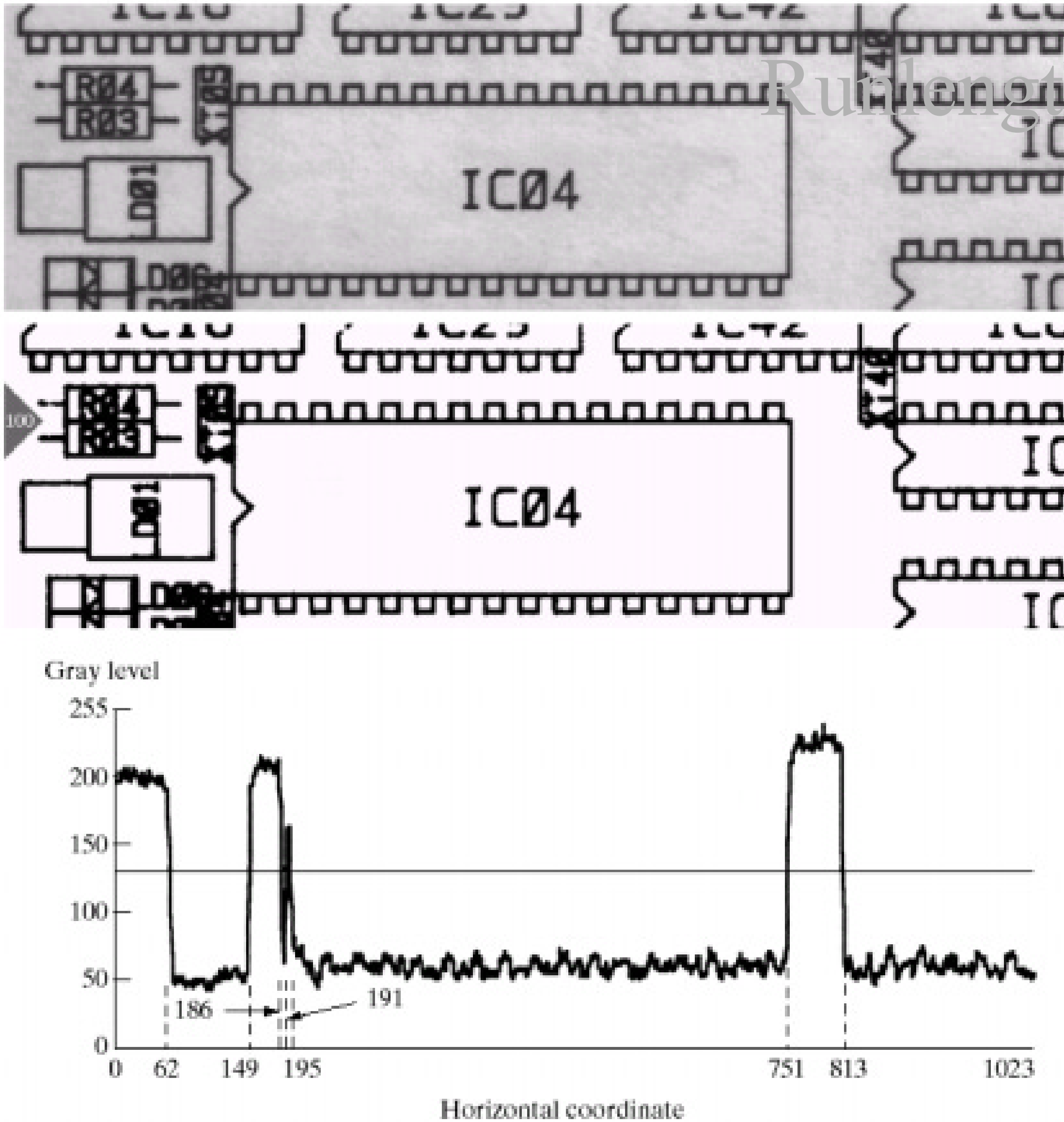
Pixel Correlations



a b
c d
e f

FIGURE 8.2 Two images and their gray-level histograms and normalized autocorrelation coefficients along one line.

Runlength Coding



a
b
c
d

FIGURE 8.3 Illustration of run-length coding: (a) original image. (b) Binary image with line 100 marked. (c) Line profile and binarization threshold. (d) Run-length code.

Line 100: (1, 63) (0, 87) (1, 37) (0, 5) (1, 4) (0, 556) (1, 62) (0, 210)

Psychovisual

- Human visual system has limitations ; good example is quantization. conveys infromation but requires much less memory/space.
- (Example: Figure 8.4 in text; matlab)

Quantization

a b c

FIGURE 8.4
(a) Original image.
(b) Uniform quantization to 16 levels. (c) IGS quantization to 16 levels.

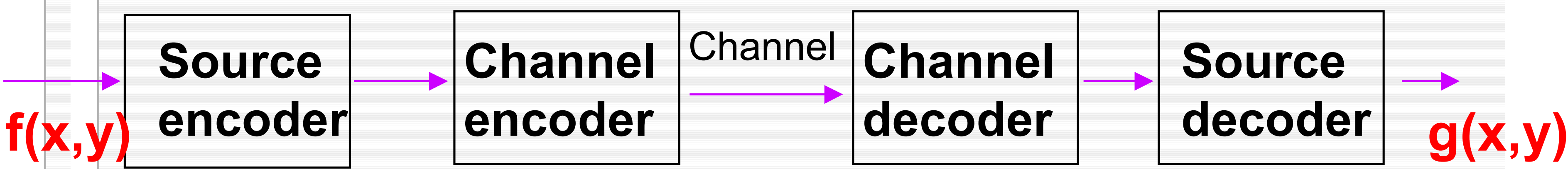
IGS Code(Table 8.2)

Pixel	Gray Level	Sum	IGS Code
$i-1$	N/A	0000 0000	N/A
i	0110 1100	0110 1100	0110
$i+1$	1000 1011	1001 0111	1001
$i+2$	1000 0111	1000 1110	1000
$i+3$	1111 0100	1111 0100	1111

TABLE 8.2
IGS quantization procedure.

General Model

General compression model



Source encoder

Source Encoder

Mapper: Designed to reduce interpixel redundancy.

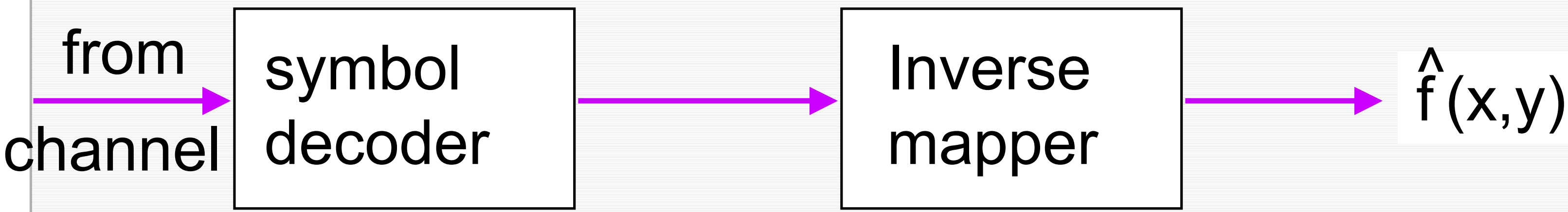
example:

- Run length encoding results in compression.
- Transfrom to another domain where the coefficients are less correlated then the original. Ex: Fourier transfrom.

Quantizer: reduces psychovisual redundancies in the image - should be left out if error-free encoding is desired.

Symbol encoder: creates a fixed/variable length code - reduces coding redundancies.

Source Decoder



Note: Quantization is **NOT** reversible

Question(s):

- Is there a minimum amount of data that is sufficient to completely describe an image without any loss of information?
- How do you measure information?

Self-Information

- Suppose an event E occurs with probability P(E) ; then it is said to contain **I (E) = -log P(E)** units of information.
- P(e) = 1 [always happens] => I(e) = 0 [conveys no information]
- If the base of the logarithm is 2, then the unit of information is called a "bit".
- If P(E) = 1/2 , $I(E) = -\log_2(1/2) = 1$ bit. Example: Flipping of a coin ; outcome of this experiment requires one bit to convey the information.

Self-Information (contd.)

Assume an information source which generates the symbols $\{a_0, a_1, a_2, \ldots, a_{L-1}\}$ with

$$prob\ \{a_i\} = p(a_i) \quad ; \quad \sum_{i=0}^{L-1} p(a_i) = 1$$

$$I(a_i) = -\log_2 p(a_i) \quad \text{bits.}$$

ENTROPY

Average information per source output is

$$H = -\sum_{i=0}^{L-1} p(a_i) \log_2 p(a_i) \quad \text{bits / symbol}$$

H is called the **uncertainity** or the **entropy** of the source.

If all the source symbols are equally probable then the source has a maximum entropy.

H gives the lower bound on the number of bits required to code a signal.

Noiseless coding theorem

(**Shannon**)
It is possible to code, without any loss of information, a source signal with entropy H bits/symbol, using H + ε bits/symbol where ε is an arbitrary small quantity.

ε can be made arbitrarily small by considering increasingly larger blocks of symbols to be coded.

Error-free coding

Error-free coding

Coding redundancy

Ex: Huffman coding

Yeilds smallest possible # of code symbols per source symbol when symbols are coded one at a time.

Interpixel redundancy

Ex: Runlength coding

Huffman code: example

Huffman code: Consider a 6 symbol source

	a_1	a_2	a_3	a_4	a_5	a_6
$p(a_i)$	0.1	0.4	0.06	0.1	0.04	0.3

Huffman coding: example (contd.)

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	$0.04_{(01011)}$				

Example (contd.)

Average length:
(0.4) (1) + 0.3 (2) + 0.1 X 3 + 0.1 X 4 + (0.06 + 0.04) 5 = 2.2 bits/symbol

$-\Sigma\ p_i \log p_i$ = 2.14 bits/symbol **(Entropy)**

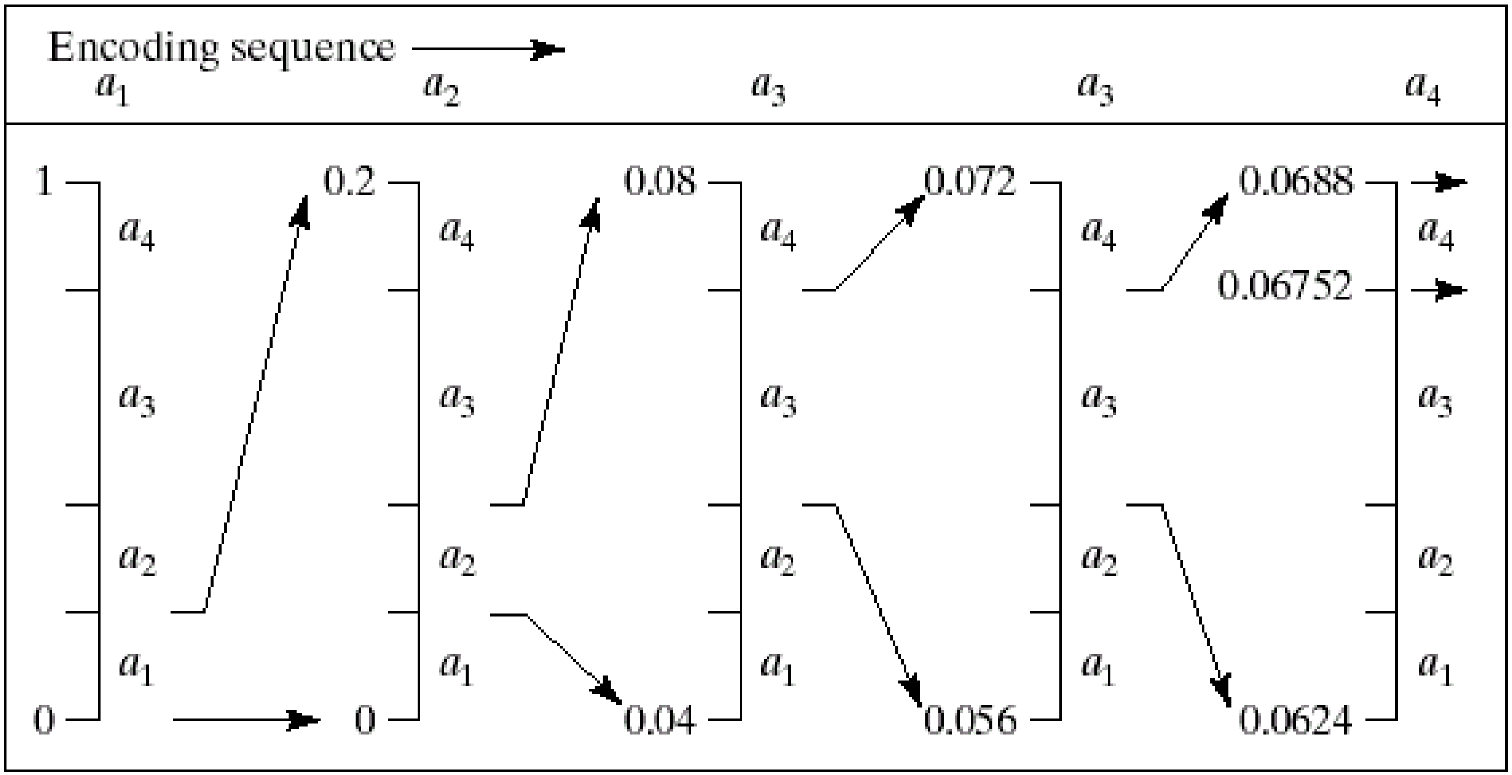
Huffman code: Steps

- Arrange symbol probabilities p_i in decreasing order
- While there is more than one node
  - Merge the two nodes with the smallest probabilities   to form a new node with probabilities equal to their sum.
  - Arbitrarily assign 1 and 0 to each pair of branches
  - merging in to a node.
- Read sequentially from the root node to the leaf node where the symbol is located.

Huffman code (final slide)

- Lossless code
- Block code
- Uniquely decodable
- Instantaneous (no future referencing is needed)

Fig. 8.13: Arithmetic Coding



Coding rate: 3/5 decimal digits/symbol
Entropy: 0.58 decimal digits/symbol.

Lempel-Ziv-Welch (LZW) coding

- Uses a dictionary
- Dictionary is adaptive to the data
- Decoder constructs the matching dictionary based on the codewords received.
- used in GIF, TIFF and PDF file formats.

LZW-an example

Consider a 4x4, 8-bits per pixel image

$$\begin{bmatrix} 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \end{bmatrix}$$

The dictionary values 0-255 correspond to the pixel values 0-255. Assume a 512 word dictionary.

LZW-dictionary construction

TABLE 8.7 LZW coding example.

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		

Run-length Coding

Run-length encoding (Binary images)

0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0

4 6 3 3 2

Lengths of 0's and 1's is encoded. Each of the bit planes in a gray scale image can be run length - encoded.

One can combine run-length encoding with variable length coding of the run-lengths to get better compression.