EE421/521 Image Processing

Lecture 12a
LOSSLESS IMAGE COMPRESSION

1

Introduction

2

The Need for Image Compression

- A 8.5x11 inch **page** scanned at 300 pixels/inch and 1 bit/pixel: 8 Mbits
- An **NTSC** (PAL) resolution color image: 720x480 (768x560) pixels/color, 8 bits/pixel, 3 color components: 8 (10) Mbits
- An **HDTV** resolution color image: 1920x1080 pixels/color, 8 bits/pixel, 3 color components: 48 Mbits
- A 35 mm **film** scanned at 12 micro meters/pixel: 3656x2664 pixels/color, 8 bits/pixel, 3 color components: 223 Mbits
- A 14x17 inch **radiograph** scanned at 70 micro meters/pixel: 5000x6000 pixels, 12 bits/pixel: 343 Mbits
- 3D **Cinema** 4k frame: 4096x2160 pixels/color, 12 bits/pixel, 3 color components, 2 views/frame: 2 Gbits
- **Dome** 8k multiview frame: 6144x6144 pixels/color, 12 bits/pixel, 3 color components, 16 views/frame: 2 Tbits

3

Benefits of Image Compression

- Less space on storage media (hard disks, CD, DVD)
- Smaller transmission bandwidth (Internet, terrestrial broadcast, satellite links, mobile phones)
- Faster downloading/uploading of multimedia files (Internet, authoring environment)

4

Why Image Compression is Possible

- In general, there are four types of redundancy in images:
  - **Spatial redundancy:** due to correlation among neighboring pixels. The amount of correlation depends on factors such as spatial resolution, bit depth, scene content, and noise
  - **Spectral redundancy:** due to correlation among the spectral (color) components (bands)
  - **Psychovisual redundancy:** due to properties of the human visual system
- Image compression aims to reduce the number of bits required to represent an image by removing one or more of the redundancies listed above.

5

Lossless vs. Lossy Image Compression

- In lossless compression, the reconstructed image is identical to the original image. Hence lossless compression is also called as noiseless or reversible.
  - Although lossless compression is desired since it results in no information loss, it provides only a moderate amount of compression.
- In lossy compression, some amount of degredation (distortion) is allowed in the re-constructed image.
  - More compression can be achieved as compared to the lossless compression at the expense of some information loss.

6

The Goal of an Image Compression Algorithm

- **LOSSLESS compression**
  - minimize the bit rate without causing any distortion.
- **LOSSY compression**
  - obtain the best possible fidelity for a given bit rate,

    or

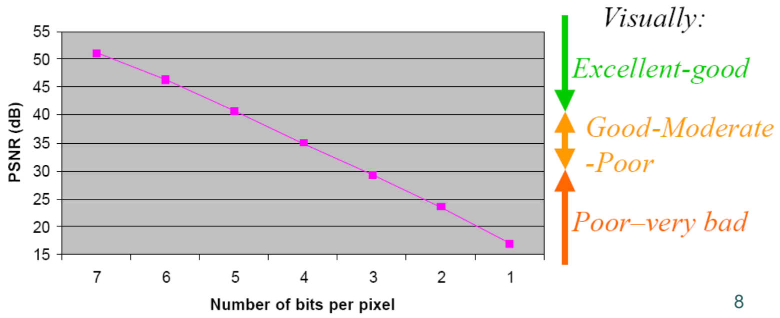
  - minimize the bit rate to achieve a given fidelity measure.

7

Quality (Fidelity) of a Compressed Image

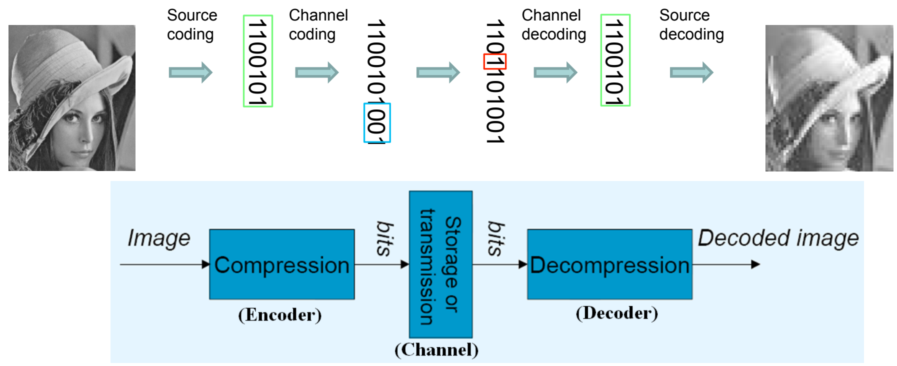
$$PSNR = \text{Peak Signal-to-Noise Ratio} =$$

$$10 \log \frac{255^2}{\sum_{all\ pixels} [X(i,j) - Y(i,j)]^2} \quad (\text{dB} = \text{decibels})$$



8

Image Compression End-To-End System



Shannon's theorem: **Source coding** (data ***compression***) and **channel coding** (error control) can be designed separately.

9

Basic Steps of Compression

Source data

↓

Energy compaction	**Transformation/Decomposition/Representation** (To put input data in a form that is amenable to eficient compression)	No compression
quality vs. compression trade-off	**Quantization** (To reduce the number of symbols representing the data)	Lossy compression (e.g., 50:1)
Encoding with 0's and 1's	**Entropy Coding** (To minimize the average length of channel symbols representing the source symbols)	Lossless compression (e.g., 2:1)

↓

Compressed bitstream

10

Amplitude Quantization (PCM) for Image Compression

$2^8 = 256$ levels

Original
(8 bits per pixel)

$2^4 = 16$ levels

2:1 compression
(4 bits per pixel)

$2^2 = 4$ levels

4:1 compression
(2 bits per pixel)

Spatial Quantization (Subsampling) for Image Compression

Original
(8 bits per pixel)

4:1 compression
(2 bits per pixel)

16:1 compression
(0.5 bits per pixel)

DCT Transformation before Amplitude Quantization

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	159	159	158	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

1260	-1	-12	-5	2	-2	-3	-1
-23	-17	-6	-3	-3	0	0	-1
-11	-9	-2	2	0	-1	-1	0
-7	-2	0	1	1	0	0	0
-1	-1	1	2	0	-1	1	1
2	0	2	0	-1	1	1	-1
-1	0	0	-1	0	2	1	-1
-3	2	-4	-2	2	1	-1	0

Energy compaction
to low frequencies

YUV Decomposition before Spatial Quantization (Subsampling)

R G B

1 + 1 + 1

8:1

R G B

$\frac{1}{8}+\frac{1}{8}+\frac{1}{8}$

8:1

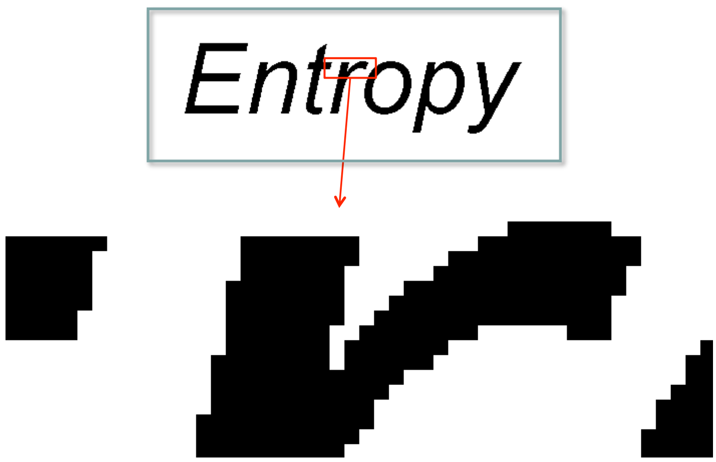
Y U V

$\frac{1}{4}+\frac{1}{16}+\frac{1}{16}$

Lossless Image Compression (Entropy Coding)

15

Binary Image Compression



16

Average Codeword Length

Symbol	Probability	Code
White	0.75	0
Black	0.25	1

$$\text{Average codeword length} = (0.75)\times 1 + (0.25)\times 1 = 1$$

No compression

17

Average Codeword Length per Symbol

2-Symbol Representation	Probability	Code
White-White	0.5	0
Black-Black	0.25	10
White-Black	0.125	110
Black-White	0.125	111

Assign shorter codewords to more probable symbols

$$\text{Average codeword length} = (0.5)\times 1 + (0.25)\times 2 + (0.125)\times 3 + (0.125)\times 3 = 1.75$$

$$\text{Average codeword length per symbol} = \frac{1.75}{2} = 0.875$$ 12.5% compression

18

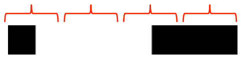
Uniquely Decodable Code

Symbol	Code
WW	0
BB	10
WB	110
BW	111

Received bitstream: $\underbrace{1\ 1\ 1}\ \underbrace{0}\ \underbrace{1\ 1\ 0}\ \underbrace{1\ 0}$

Decoded symbols: BW WW WB BB

Decompressed image:



$$\text{Average codeword length per symbol} = \frac{1.75}{2} = 0.875 \Rightarrow$$ 12.5% compression

19

Source Reduction

Original source		Source reduction			
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				

Combine the two lowest probability symbols until having two symbols only

20

Huffman Coding

Original source			Source reduction							
Symbol	Probability	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								

$$\text{Average codeword length} = \sum_n p(a_n)l(a_n) = 2.2$$

Split the symbols and append 0 and 1 to their respective codewords

21

Huffman Coding

- Uniquely decodable
  - No codeword is a prefix to another code word
- Results in the lowest average codeword length
  - Variable length code (VLC)

Code
1
00
011
0100
01010
01011

22

Entropy of the Source

$$H = \sum_n p(a_n) \log \frac{1}{p(a_n)}$$

$\log_2 \frac{1}{p(a_n)}$: Information provided by symbol a_n in units of bits

H: Total information content of the source measured in bits

23

Fixed Length Coding

Symbol	Probability	Code
a_1	0.25	00
a_2	0.25	10
a_3	0.25	01
a_4	0.25	11

Average codeword length is 2; entropy of the source is 2.

- Fixed-length coding is optimal (the entropy of the source equal to the fixed codeword length) only if:
  1) the number of symbols is equal to a power of 2, and
  2) all the symbols are equiprobable.

24

Entropy is the Lower Bound on Average Codeword Length

Probability	Code
0.4	1
0.3	00
0.1	011
0.1	0100
0.06	01010
0.04	01011

$$\boxed{H \le \bar{l}_{\text{Huffman}} \le \bar{l}_{\text{Any}}}$$

Fixed Length Coding:

6 symbols $\Rightarrow$ 3 bits per symbol

Huffman Coding:

$$\bar{l} = \sum_n p(a_n) l(a_n) = 2.2$$

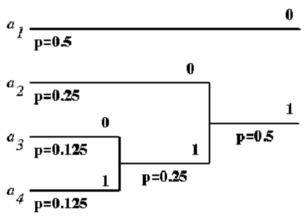
Entropy of the source:

$$H = \sum_n p(a_n) \log \frac{1}{p(a_n)} = 2.1435...$$

25

Example: Probabilities are Negative Powers of 2

Symbol	*Probability*	*Information*
a_1	0.50	1 bit
a_2	0.25	2 bits
a_3	0.125	3 bits
a_4	0.125	3 bits



The average codeword length is

R = 0.5 × 1 + 0.25 × 2 + (0.125 × 3) × 2 = 1.75

The entropy of the source is

$H = -0.5 \log_2 0.5 - 0.25 \log_2 0.25 - (0.125 \log_2 0.125) \times 2 = 1.75$

- Huffman coding achieves the entropy of the source when the symbol probabilities are powers of 2.

26

Entropy Indicates an Upper Bound on ACL as well

$$\boxed{H(S) \le \bar{l}_{\text{Huffman}} < H(S) + 1}$$

Probability	Huffman Code	Information	Next Nearest Integer	Sub-optimal Code
0.4	1	1.32	2	11
0.3	00	1.74	2	10
0.1	011	3.22	4	0001
0.1	0100	3.22	4	0010
0.06	01010	4.06	5	00110
0.04	01011	4.64	5	00111
$H = 2.14$	$\bar{l} = 2.2$			$\bar{l} = 2.7$

27

Joint and Conditional Entropy

Let X and Y be two consequtive symbols. Then we have

Joint entropy ⟹ $H(X,Y) \le H(X) + H(Y)$ ⟹ Vector coding

Conditional entropy ⟹ $H(X|Y) \le H(X)$ ⟹ Differential coding

with equality if and only if X and Y are independent.

28

Advantages of Vector Coding

$$H(X_1,\ldots,X_N) \le \bar{l}(X_1,\ldots,X_N) < H(X_1,\ldots,X_N)+1$$

1

Smallest achievable average codeword length is closer to the theoretical minimum

$$\frac{1}{N}H(X^N) \le \frac{1}{N}\bar{l}(X^N) < \frac{1}{N}H(X^N)+\frac{1}{N}$$

$$\frac{1}{N}H(X^N) \le H(X)$$

2

Reduced theoretical minimum *if source symbols are dependent*

29

Advantage of Conditional Coding

$$H(X_1 \mid X_2,\ldots,X_N) \le \bar{l}(X_1 \mid X_2,\ldots,X_N) < H(X_1 \mid X_2,\ldots,X_N)+1$$

$$H(X_1 \mid X_2,\ldots,X_N) \le H(X_1)$$

1

Reduced theoretical minimum *if source symbols are dependent*

30

Summary

- Scalar coding: $\boxed{H_1(S) \le L < H_1(S)+1}$
  - Assign one codeword to one symbol at a time
  - Problem: Could differ from the entropy by up to 1 bit/symbol
- Vector coding:
  - Assign one codeword for each group of N symbols
  - Larger N → Lower rate, but higher complexity

$$\underbrace{H_N(S) \le L_N}_{\text{Average minimal number of bitsper vector}} < H_N(S)+1 \quad \Rightarrow \quad \boxed{\frac{1}{N}H_N(S) \le \underbrace{L}_{\text{Average minimal number of bits per sample (bit rate)}} < \frac{1}{N}H_N(S)+\frac{1}{N}}$$

- Conditional coding (context-based coding)
  - The codeword for the current symbol depends on the pattern (context) formed by the previous M symbols

$$\boxed{H_{C,M}(S) \le L_{C,M} \le H_{C,M}(S)+1} \longrightarrow$$

Average minimal number of bits per sample with a conditioning order M

31

Example

- Four symbols: “a”, “b”, “c”, “d”
- Symbol probabilities (pmf): $\mathbf{p}^T = [0.5000, 0.2143, 0.1703, 0.1154]$
- 1st order conditional pmf: $p(c\,|\,b)$

$$p(b\,|\,a) \longrightarrow \begin{bmatrix} 0.6250 & 0.3750 & 0.3750 & 0.3750 \\ 0.1875 & 0.3125 & 0.1875 & 0.1875 \\ 0.1250 & 0.1875 & 0.3125 & 0.1250 \\ 0.0625 & 0.1250 & 0.1250 & 0.3125 \end{bmatrix} \longleftarrow p(a\,|\,d)$$

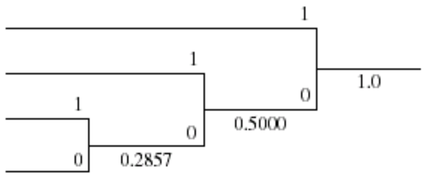
- 2nd order vector pmf: $p(f_{n-1}, f_n) = p(f_{n-1})q(f_n \,|\, f_{n-1})$.

  E.g., $p(ab) = p(a)q(b\,|\,a) = 0.5 * 0.1875 = 0.0938$

32

Scalar Huffman Coding

Symbol	Probability	Codeword	Codeword length
"a"	0.5000	"1"	1
"b"	0.2143	"01"	2
"c"	0.1703	"001"	3
"d"	0.1154	"000"	3



Bit rate $R = 1.7857$ Entropy $H_1 = 1.7707$

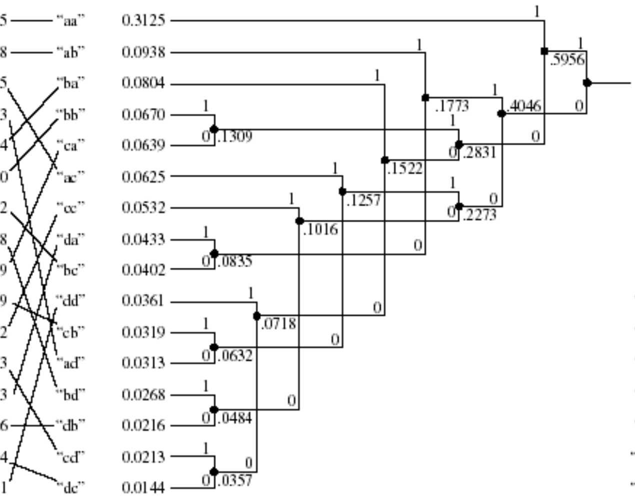
$H_1 \le R < H_1 + 1$ → Bound for scalar lossless coding

$1.7707 \le 1.7857 < 2.7707$ ✓

33

Vector Huffman Coding

Symbol	Probability	Reordered symbol	Probability	Codeword	Length
"aa"	0.3125	"aa"	0.3125	"11"	2
"ab"	0.0938	"ab"	0.0938	"011"	3
"ac"	0.0625	"ba"	0.0804	"1001"	4
"ad"	0.0313	"bb"	0.0670	"1011"	4
"ba"	0.0804	"ca"	0.0639	"1010"	4
"bb"	0.0670	"ac"	0.0625	"0011"	4
"bc"	0.0402	"cc"	0.0532	"0001"	4
"bd"	0.0268	"da"	0.0433	"0101"	4
"ca"	0.0639	"bc"	0.0402	"0100"	4
"cb"	0.0319	"dd"	0.0361	"10001"	5
"cc"	0.0532	"cb"	0.0319	"00101"	5
"cd"	0.0213	"ad"	0.0313	"00100"	5
"da"	0.0433	"bd"	0.0268	"00001"	5
"db"	0.0216	"db"	0.0216	"00000"	5
"dc"	0.0144	"cd"	0.0213	"100001"	6
"dd"	0.0361	"dc"	0.0144	"100000"	6



$R^2 = 3.5003$ $H_2 = 3.4629$

Vector Huffman Coding

$$\text{bit rate per sample: } R_2 = L = \frac{3.5003}{2} = 1.75015$$

$$\text{entropy per sample: } \frac{H_2}{2} = 1.7314$$

$$\text{Bound for vector lossless coding: } \frac{1}{2}H_2 \le R_2 < \frac{1}{2}H_2 + \frac{1}{2}$$

$$1.7314 \le 1.75015 < 2.2314(vector)$$

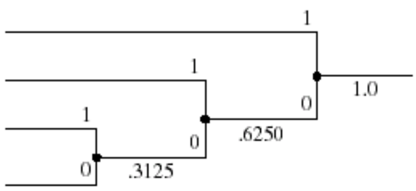
smaller

$$1.7707 \le 1.7857 < 2.7707(scalar)$$

35

Conditional Huffman Coding

Symbol	Probability	Codeword	Length
"a"/"b"	0.3750	"1"	1
"b"/"b"	0.3125	"01"	2
"c"/"b"	0.1875	"001"	3
"d"/"b"	0.1250	"000"	3



$R_{C,\text{"b"}} = 1.9375 \qquad H_{C,\text{"b"}} = 1.8829$

$$R_{C,\text{"a"}} = 1.5625, R_{C,\text{"b"}} = R_{C,\text{"c"}} = R_{C,\text{"d"}} = 1.9375,$$
$$R_{C,1} = 0.5 \times 1.5625 + 0.5 \times 1.9375 = 1.7500$$

$$H_{C,\text{"a"}} = 1.5016, H_{C,\text{"b"}} = H_{C,\text{"c"}} = H_{C,\text{"d"}} = 1.8829,$$
$$H_{C,1} = 0.5 \times 1.5016 + 0.5 \times 1.8829 = 1.6922$$

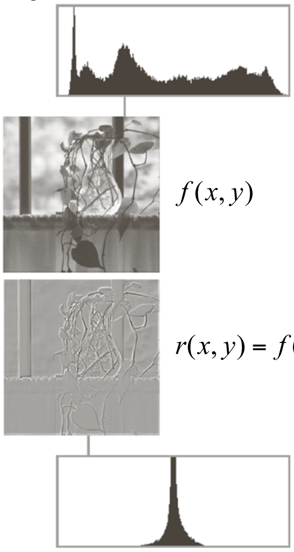
Bound for Conditional Lossless Coding

$$H_{C,1} \le R_{C,1} \le H_{C,1} + 1$$

$$1.6922 < 1.7500 < 2.6922$$

36

Differential Coding



- The probabilities of the symbols are appoximated by the histogram of the image data
- Residual image has a peaked histogram, hence it can be compressed more easily
- Approximates conditional coding

37

Lossless Predictive Coding (DPCM)

- The new information in a pixel is defined as the difference between the actual value and a prediction of that pixel based on a set of its previously coded neighbors.
- Examples of predictors:

$$\hat{x} = \text{int}\left(\frac{a+c}{2}\right)$$

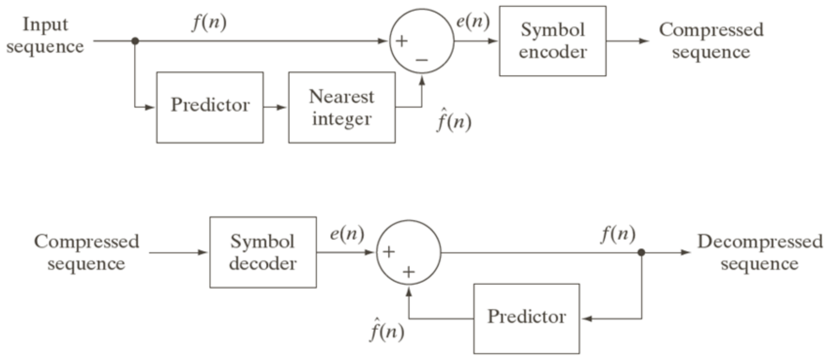
$$\hat{x} = \text{int}\left(\frac{a+b+c+d}{4}\right)$$

b	c	d	
a	**x**		

The prediction is rounded to the nearest integer to ensure an integer prediction error.

- The predictive transform removes spatial redundancy.

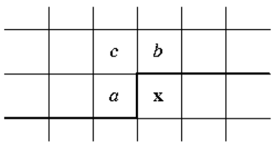
Encoder/Decoder Structure for Predictive Coding



39

JPEG Lossless Coding

- The lossless mode of operation of the original JPEG standard uses integer-predictive coding.



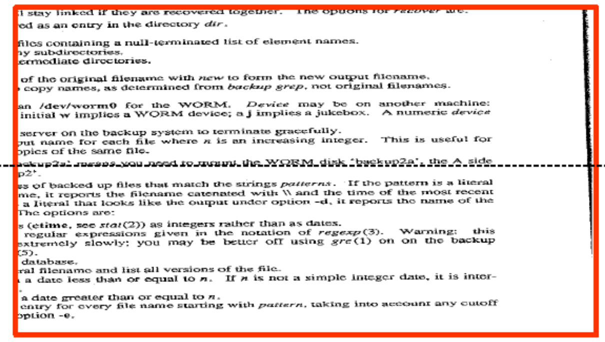
Selection	Predictor
0	No prediction
1	a
2	b
3	c
4	a+b-c
5	a-(b-c)/2
6	b-(a-c)/2
7	(a+b)/2

- The first line of image is always coded using selection 1.
- The symbols for the prediction errors are VLC coded.

40

Run-Length Coding

- Useful for coding of “long runs of the same symbol”
- Example: Binary picture (fax)



l stay linked if they are recovered together. The options for *recover* are:
ed as an entry in the directory *dir*.

files containing a null-terminated list of element names.
ny subdirectories.
ermediate directories.

of the original filename with *new* to form the new output filename.
copy names, as determined from *backup grep*, not original filenames.

an /dev/worm0 for the WORM. *Device* may be on another machine:
initial w implies a WORM device; a j implies a jukebox. A numeric *device*

server on the backup system to terminate gracefully.
put name for each file where *n* is an increasing integer. This is useful for
opies of the same file.
ackup2a' means you need to mount the WORM disk 'backup2a', the A side
p2'.
es of backed up files that match the strings *patterns*. If the pattern is a literal
me, it reports the filename catenated with \\ and the time of the most recent
a literal that looks like the output under option -d, it reports the name of the
The options are:
s (ctime, see *stat*(2)) as integers rather than as dates.
regular expressions given in the notation of *regexp*(3). Warning: this
extremely slowly; you may be better off using *gre*(1) on on the backup
(5).
database.
ral filename and list all versions of the file.
a date less than or equal to *n*. If *n* is not a simple integer date, it is inter-

a date greater than or equal to *n*.
entry for every file name starting with *pattern*, taking into account any cutoff
option -e.

41

Run-Length Coding of Binary Images

- Representing
  - Black = “1”
  - White = “0”

  we get for instance

  “0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 …”

- We now code the “runs” of identical symbols
  - **8** (“zeroes”) **2** (“ones”) **6** (“zeroes”) **5** (“ones”) …
- The run lengths themselves need to be encoded by bit patterns
  - For instance Huffman code

42

Run-Length Coding

- Each line of the image is encoded as a series of VLC that represent the run-lengths of alternating white and black runs in a left-to-right scan.
- Since the statistics of white and black runs are different, we have different codebooks to represent the white and black run-lengths.

ITU-T Terminating Codes

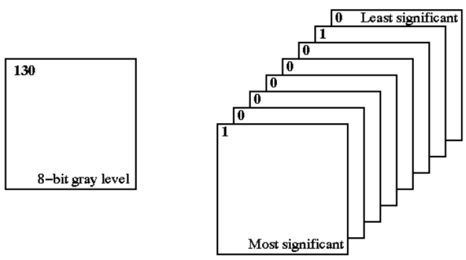
Run Length	*White Codeword*	*Black Codeword*
0	00110101	0000110111
⋮	⋮	⋮
63	00110100	000001100111

- The symbol probabilities were obtained from a number of test documents both typed and handwritten in several languages.

43

Bit-Plane Run-Length Coding

- *Bit-Plane Decomposition*: Decompose a multilevel (monochrome or color) image into a series of binary images.



Bit-plane decomposition of an 8-bit image.

The levels of an m-bit gray-scale image can be represented as

$$a_{m-1}\, 2^{m-1} + a_{m-2}\, 2^{m-2} + \ldots + a_1\, 2 + a_0\, 2^0$$

where a_i, i=0,…, m-1 are either 0 or 1.

44

Zero-Run Coding

- We will see that efficient "transforms" used in compression produce
  - A lot of "zero" values
  - Some (significant) non-zero values

0	2	0	-1
0	0	-1	0
2	-4	-2	2

- Typical symbol sequences to be coded:

"5 1 0 0 0 0 0 0 0 3 0 0 6 0 0 0 0 1 0 0 0 0 …"

  - Will be done by {zero-run, non-zero symbol} pairs
  - Here: "{0,5}, {0,1}, {7,3}, {2,6}, {4,1}, …"
  - The pairs will now be assigned a Huffman code

45

Project 3.3

Compression

Due 12.01.2014

46

Problem 3.3 Entropy

1. Select a 256x256 monochrome image and display it.
2. Find the histogram of the image.
3. Normalize the histrogram to obtain the pdf of the image.
4. Calculate the entropy of the image using its pdf.
5. Obtain a difference image by taking first-order horizontal differences:

$$d(m,n) = x(m,n) - x(m-1,n)$$

6. Find the histogram of the difference image.
7. Normalize the histrogram to obtain the pdf of the difference image.
8. Calculate the entropy of the difference image using its pdf.
9. Compare the entropies of the original and difference images and comment on the difference.

47

Next Lecture

- LOSSY IMAGE COMPRESSION

48