
525.749 Image Compression and Packet Video

Dr. Nicholas Beser

E-mail: Nicholas.Beser@jhuapl.edu

11/01/04

Lecture 7 - Foundation of Video Coding

Part I: Overview and Binary Encoding

<http://www.apl.jhu.edu/Notes/Beser/525759/index.html>

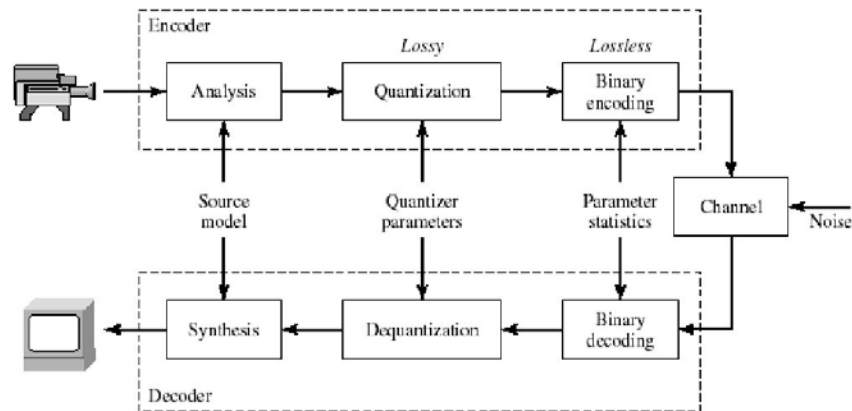
1

Outline

- Overview of video coding systems
- Review of probability and information theory concepts
- Binary encoding
 - Information theory bounds
 - Huffman coding
 - Arithmetic coding

2

Components in a Coding System



Video Coding Techniques Based on Different Source Models

TABLE 8.1 COMPARISON OF SOURCE MODELS, PARAMETER SETS, AND CODING TECHNIQUES.

Source model	Encoded parameters	Coding technique
Statistically independent pels	Color of each pel	PCM
Statistically dependent pels	Color of each block	Transform coding, predictive coding, and vector quantization
Translationally moving blocks	Color and motion vector of each block	Block-based hybrid coding Waveform-based techniques
Moving unknown objects	Shape, motion, and color of each object	Analysis-synthesis coding
Moving known object	Shape, motion, and color of each known object	Knowledge-based coding
Moving known object with known behavior	Shape, color, and behavior of each object	Semantic coding Content-dependent-techniques

Statistical Characterization of Random Sources

- **Source: a random sequence (discrete time random process),** $F = \{F_n\}$
 - Ex 1: an image that follows a certain statistics
 - » F_n represents the *possible value* of the n-th pixel of the image, $n=(m,n)$
 - » f_n represents the *actual value* taken
 - Ex 2: a video that follows a certain statistics
 - » F_n represents the *possible value* of the n-th pixel of a video, $n=(k,m,n)$
 - » f_n represents the *actual value* taken
 - Continuous source: F_n takes continuous values (analog image)
 - Discrete source: F_n takes discrete values (digital image)
- **Stationary source: statistical distribution invariant to time (space) shift**
- **Probability distribution**
 - probability mass function (pmf) or probability density function (pdf): $p_{F_n}(f) \quad p(f)$
 - Joint pmf or pdf: $p_{F_{n+1}, F_{n+2}, \dots, F_{n+N}}(f_1, f_2, \dots, f_N) \quad p(f_1, f_2, \dots, f_N)$
 - Conditional pmf or pdf:

$$p_{F_n}(F_{n-1}F_{n-2}, \dots, F_{n-M} | f_{M+1}, f_{M-1}, \dots, f_1) \quad p(f_{M+1} | f_M, f_{M-1}, \dots, f_1)$$

Information Content Characterization of Discrete RVs

- **Entropy of one RV:** $H(F) = - \sum_{f \in A} p_F(f) \log_2 p_F(f)$
 - Entropy is a measure of uncertainty or information content
 - Very uncertain -> high information content
- **Joint entropy of two RVs:** $H(f, g) = - \sum_{f \in A_f} \sum_{g \in A_g} p_{F,G}(f, g) \log_2 p_{F,G}(f, g)$
 - Uncertainty of two RVs together
 - $H(F, G) = H(G) + H(F|G) \quad H(F, G) \leq H(F) + H(G)$
- **Conditional entropy between two RVs:** $H(F|G) = \sum_{g \in A_g} p_G(g) H(F|g)$
 - Uncertainty of one RV given the other RV
 - $H(F) \geq H(F|G)$
- **Mutual information between two RVs :** $I(F; G) = \sum_{f \in A_f} \sum_{g \in A_g} p_{F,G}(f, g) \log_2 \frac{p_{F,G}(f, g)}{p_F(f)p_G(g)}$
 - Information provided by G about F
 - $I(F; G) = H(F) - H(F|G) \quad I(F; G) \leq H(F) \quad I(F; G) = H(F) + H(G) - H(F, G)$

Information Content Characterization of Discrete Sources

- **N-th order entropy**

$$\begin{aligned}
 H_N(F) &= H(F_1, F_2, \dots, F_N) \\
 &= - \sum_{[f_1, f_2, \dots, f_N] \in A^N} p(f_1, f_2, \dots, f_N) \log_2 p(f_1, f_2, \dots, f_N)
 \end{aligned}$$

- **M-th order conditional entropy**

$$\begin{aligned}
 H_{C,M}(F) &= H(F_{M+1} | F_M, F_{M-1}, \dots, F_1) \\
 &= \sum_{[f_1, f_2, \dots, f_M] \in A^M} p(f_1, f_2, \dots, f_M) H(F_{M+1} | f_M, f_{M-1}, \dots, f_1) \\
 &= \sum_{f_{M+1} \in A} p(f_{M+1} | f_M, f_{M-1}, \dots, f_1) \log_2 p(f_{M+1} | f_M, f_{M-1}, \dots, f_1) \\
 H(F) &\leq H_{C,N-1}(F) \leq \frac{1}{N} H_N(F) \leq H_1(F)
 \end{aligned}$$

7

Information Content Characterization of Discrete Sources

- **Entropy rate (lossless coding bound)**

$$H(F) = \lim_{N \rightarrow \infty} \frac{1}{N} H_N(F) = \lim_{N \rightarrow \infty} \frac{1}{N} H_{C,N}(F)$$

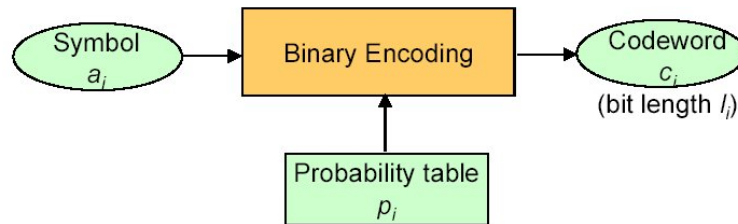
- **N-th order mutual information (lossy coding bound)**

$$\begin{aligned}
 I_N(F; G) &= \\
 &= \sum_{[f_1, f_2, \dots, f_N] \in A_f^N} \sum_{[g_1, g_2, \dots, g_N] \in A_g^N} p(f_1, f_2, \dots, f_N, g_1, g_2, \dots, g_N) \\
 &\bullet \log_2 \frac{p(f_1, f_2, \dots, f_N, g_1, g_2, \dots, g_N)}{p(f_1, f_2, \dots, f_N) p(g_1, g_2, \dots, g_N)}
 \end{aligned}$$

8

Lossless Coding (Binary Encoding)

- Binary encoding is a necessary step in any coding system
 - Applied to
 - » original symbols (e.g. image pixels) in a discrete source,
 - » or converted symbols (e.g. quantized transformed coefficients) from a continuous or discrete source
- Binary encoding process (scalar coding)



Bit rate (Bit/symbol):
$$R = \sum_{a_i \in A} p(a_i)l(a_i)$$

9

Bound for Lossless Coding

- Scalar coding: $H_1(F) \leq R_1(F) \leq H_1(F) + 1$
 - 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

$$H_N(F) \leq R^N(F) \leq H_N(F) + 1 \quad H_N(F)/N \leq \bar{R}_N(F) \leq H_N(F)/N + 1/N.$$

$$\lim_{N \rightarrow \infty} \bar{R}_N(F) = \bar{H}(F)$$

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

$$H_{C,M}^m(F) \leq \bar{R}^N(F) \leq H_{C,M}^m(F) + 1. \quad H_{C,M}(F) \leq \bar{R}_{C,M}(F) \leq H_{C,M}(F) + 1.$$

$$\bar{H}(F) \leq \lim_{M \rightarrow \infty} \bar{R}_{C,M}(F) \leq \bar{H}(F) + 1.$$

10

Binary Encoding: Requirement

- A good code should be:
 - Uniquely decodable
 - Instantaneously decodable – prefix code

Codebook 1 (a prefix code)	Codebook 2 (not a prefix code)																				
<table border="1" style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="text-align: center;">Symbol</th> <th style="text-align: center;">Codeword</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">a_1</td> <td style="text-align: center;">"0"</td> </tr> <tr> <td style="text-align: center;">a_2</td> <td style="text-align: center;">"10"</td> </tr> <tr> <td style="text-align: center;">a_3</td> <td style="text-align: center;">"110"</td> </tr> <tr> <td style="text-align: center;">a_4</td> <td style="text-align: center;">"111"</td> </tr> </tbody> </table>	Symbol	Codeword	a_1	"0"	a_2	"10"	a_3	"110"	a_4	"111"	<table border="1" style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="text-align: center;">Symbol</th> <th style="text-align: center;">Codeword</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">a_1</td> <td style="text-align: center;">"0"</td> </tr> <tr> <td style="text-align: center;">a_2</td> <td style="text-align: center;">"01"</td> </tr> <tr> <td style="text-align: center;">a_3</td> <td style="text-align: center;">"100"</td> </tr> <tr> <td style="text-align: center;">a_4</td> <td style="text-align: center;">"011"</td> </tr> </tbody> </table>	Symbol	Codeword	a_1	"0"	a_2	"01"	a_3	"100"	a_4	"011"
Symbol	Codeword																				
a_1	"0"																				
a_2	"10"																				
a_3	"110"																				
a_4	"111"																				
Symbol	Codeword																				
a_1	"0"																				
a_2	"01"																				
a_3	"100"																				
a_4	"011"																				

Bitstream: 0 0 1 1 0 1 0 1 1 0 1 0 0

Decoded string based on codebook 1: 0|0|1 1|0|1 0|1 1|0|1 0|0 → $a_1 a_1 a_3 a_2 a_3 a_2 a_1$
 (can decode instantaneously)

Decoded string based on codebook 2: 0|0 1 1|0 1|0 1 1|0|1 0 0 → $a_1 a_1 a_2 a_4 a_1 a_3$
 (must look ahead to decode)

Huffman Coding

- **Idea:** more frequent symbols -> shorter codewords
- **Algorithm:**
 - Step 1: Arrange the symbol probabilities $p(a_l)$, $l = 1, 2, \dots, L$, in a decreasing order and consider them as leaf nodes of a tree.
 - Step 2: While there is more than one node:
 - (a) Find the two nodes with the smallest probability and arbitrarily assign 1 and 0 to these two nodes.
 - (b) Merge the two nodes to form a new node whose probability is the sum of the two merged nodes. Go back to Step 1.
 - Step 3: For each symbol, determine its codeword by tracing the assigned bits from the corresponding leaf node to the top of the tree. The bit at the leaf node is the last bit of the codeword.
- **Huffman coding generate prefix code.**
- **Can be applied to one symbol at a time (scalar coding), or a group of symbols (vector coding), or one symbol conditioned on previous symbols (conditional coding)**

Huffman Coding Example: Source description

- Four symbols: "a","b","c","d"
- pmf: $p^T = [0.5000, 0.2143, 0.1703, 0.1154]$

- 1st order conditional pmf:

$$[Q] = \begin{bmatrix} 0.6250 & 0.3750 & 0.3750 & 0.3750 \\ 0.1875 & 0.3125 & 0.1875 & 0.1875 \\ 0.1250 & 0.1875 & 0.3125 & 0.1250 \\ 0.0625 & 0.1250 & 0.1250 & 0.3125 \end{bmatrix}$$

- 2nd order pmf:

$$p(f_{n-1}, f_n) = p(f_{n-1})q(f_n|f_{n-1}).$$

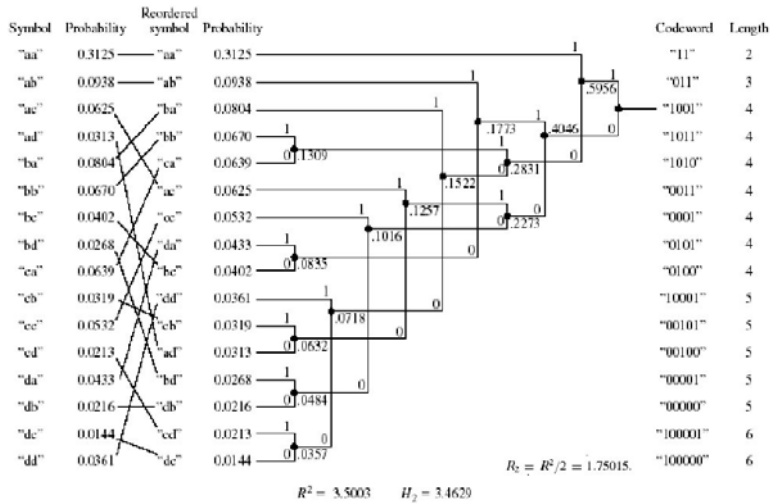
$$\text{Ex. } p("ab") = p("a")q("b"/"a") = 0.5 * 0.1875 = 0.0938$$

Huffman Coding Example: Scalar 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

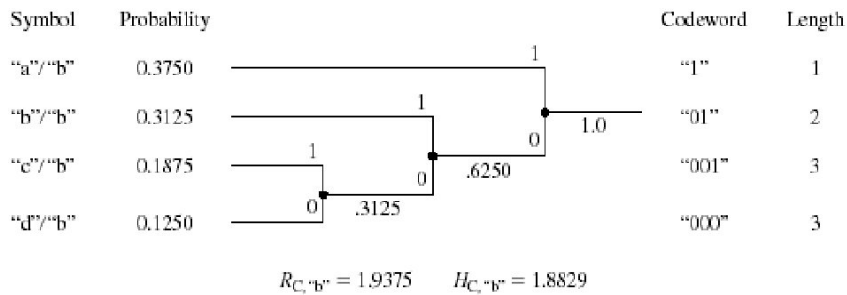
Bit rate $R = 1.7857$ Entropy $H_1 = 1.7707$

Huffman Coding Example: Vector Coding



15

Huffman Coding Example: Conditional Coding



$$R_{C,"a"} = 1.5625, R_{C,"b"} = R_{C,"c"} = R_{C,"d"} = 1/9375, R_{C,1} = 1.7500$$

$$H_{C,"a"} = 1.5016, H_{C,"b"} = H_{C,"c"} = H_{C,"d"} = 1.8829, H_{C,1} = 1.6922$$

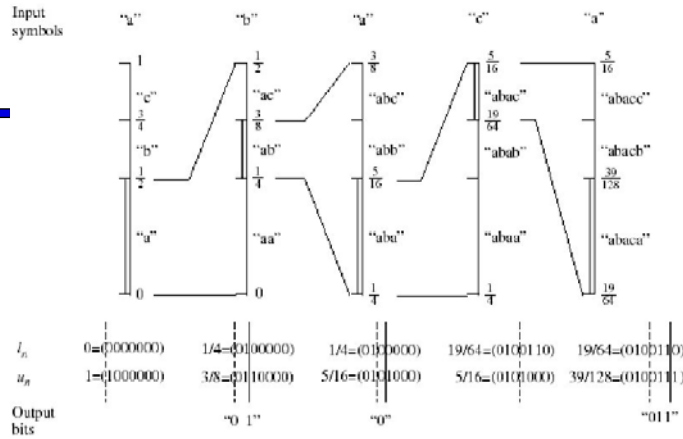
16

Arithmetic Coding

- **Basic idea:**
 - Represent a sequence of symbols by an interval with length equal to its probability
 - The interval is specified by its lower boundary (l), upper boundary (u) and length d (=probability)
 - The codeword for the sequence is the common bits in binary representations of l and u
- **The interval is calculated sequentially starting from the first symbol**
 - The initial interval is determined by the first symbol
 - The next interval is a subinterval of the previous one, determined by the next symbol

$$d_n = d_{n-1} * p_l; \quad I_n = I_{n-1} + d_{n-1} * q_{l-1}; \quad u_n = I_n + d_n.$$

Encoding:



Decoding:

Received bits	Interval	Decoded symbol
"0"	[0, 1/2)	"a"
"01"	[1/4, 1/2)	—
"010"	[1/4, 3/8)	"b"
"0100"	[1/4, 5/16)	"a"
"01001"	[9/32, 5/16)	—
"010011"	[19/64, 5/16)	"c"
...

Huffman vs. Arithmetic Coding

- **Huffman coding**
 - Convert a fixed number of symbols into a variable length codeword
 - Efficiency:

$$H_N(F)/N \leq \bar{R}_N(F) \leq H_N(F)/N + 1/N.$$
 - To approach entropy rate, must code a large number of symbols together
 - Used in all image and video coding standards
- **Arithmetic coding**
 - Convert a variable number of symbols into a variable length codeword
 - Efficiency:

$$H_N(F)/N \leq R \leq H_N(F)/N + 2/N. \quad N \text{ is sequence length}$$
 - Can approach the entropy rate by processing one symbol at a time
 - Easy to adapt to changes in source statistics
 - Integer implementation is available
 - Used as advanced options in image and video coding standards

19

Summary

- **Coding system:**
 - original data -> model parameters -> quantization-> binary encoding
 - Waveform-based vs. content-dependent coding
- **Lossless coding**
 - Bit rate bounded by entropy rate of the source
 - Huffman coding:
 - » Scalar, vector, conditional coding
 - » can achieve the bound only if a large number of symbols are coded together
 - » Huffman coding generates prefix code (instantaneously decodable)
 - Arithmetic coding
 - » Can achieve the bound by processing one symbol at a time
 - » More complicated than scalar or short vector Huffman coding

20

Homework 5

- **Reading assignment:**
 - Sec. 8.1-8.5 (excluding Sec. 8.3.2)
- **Written assignment**
 - Prob. 8.6,8.7
 - MATLAB assignment. Code the arithmetic coder described in Example 8.4 in MATLAB (You can also do problem 8.7 using it). Adjust the algorithm to support more than three symbols.
 - » Use the occurrence frequency of each symbol in the sequence as the estimate of the probability of the symbol:
 - Source Sequence: aacbaabaabaaabed
 - » Compare the result to scalar Huffman coding
 - Comment on the size of the code
 - Comment on the efficiency of the Huffman symbol “d” compared to the arithmetic encode
 - Do the same for the symbol “a”
 - » How far apart are the lower and upper limits after the last symbol