

Image Compression and Packet Video




- **Instructor:**
 - **Will Geckle - Johns Hopkins University/Applied Physics Lab**
 - **Telephone:**
 - » (240) 228-6684 (Washington), or
 - » (443) 778-6684 (Baltimore)
 - » (240) 228-6583 or (443) 778-6583 (FAX)
 - **Internet E-mail Address:**
 - » **william.geckle@jhuapl.edu**
 - **Class Website:**
 - » <http://aplcnmp.apl.jhu.edu/Notes/Geckle/525759>

Image Compression and Packet Video




- **Textbook: Introduction to Data Compression, Khalid Sayood.**
- **Excellent: Many examples and fairly easy to read.**
- **Newsgroups: comp.compression**
good source of discussion about current compression algorithms and standards with a few bizarre questions also. Read it now and after course is over to mark your progress.


Class Survey

- 
- **Name**
 - **Employer**
 - **Work Phone Number**
 - **Home Phone Number**
 - **E-Mail Address (if you have one)**
 - **Any Data Compression, Signal or Image Processing Background?**
 - **Any Matlab background?**
 - **Do you have high speed internet access?**
 - **Can you print out viewgraphs via Adobe Acrobat?**


Goals Of The Class

- 
- **Course provides a fundamental understanding of the techniques used to design still and moving image compression systems.**
 - **Cover underlying problem:**
 - Data Characteristics
 - Design Trades (Sensor to Display)
 - Transformation, Quantization and Coding
 - **Standards will be addressed in terms of both Compression and Communication Protocols**


Class Organization

- 
- **Class is organized along the lines of a Seminar**
 - **I have lectures planned on Image Coding, Lossless Coding, Predictive Coders, Transform Based Coders, Video Coding and Communication Networks**
 - **Instead of homeworks and tests, the class will have up to nine MATLAB based assignments.**
 - Each assignment will have one extra credit section
 - **Final Grade will be dependent on how many assignments are completed.**

MATLAB (for PC/Windows)

- 
- **The University has purchased Matlab 5.3 for the IBM PC/Windows**
 - **The following Add-ons/Toolboxes were also purchased:**
 - Signal Processing Toolbox
 - Image Processing Toolbox
 - Wavelet Toolbox
 - **Grading Criteria:**
 - Pass take-home, open book final + ...
 - Six of the Nine Lab Projects - C
 - At Least Eight Lab Projects and two extra credit - B
 - At least six of the extra credit parts plus the Nine Labs – A
 - Reduce grade by one for failing final but meeting other criteria

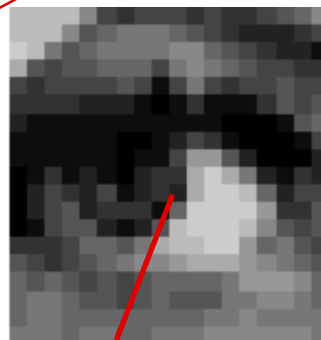
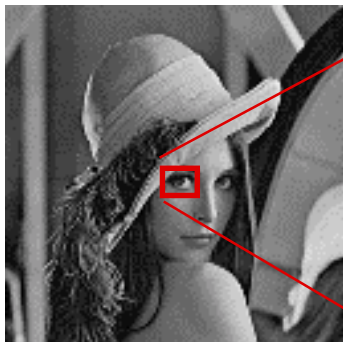
Lab Subjects

- 
- **Image display and measures, Image Processing basics**
 - **Subjective Quality Assessment**
 - **Numerical Quantization Theory**
 - **Coding methods**
 - **Image Transforms, and their statistical properties**
 - **JPEG and the Image Quality Experiment**
 - **Vector Quantization**
 - **Motion Estimation Experiment**
 - **MPEG-4 Concepts (Image Mosaicing from MPEG video)**

Digital Image Formation



Source Image



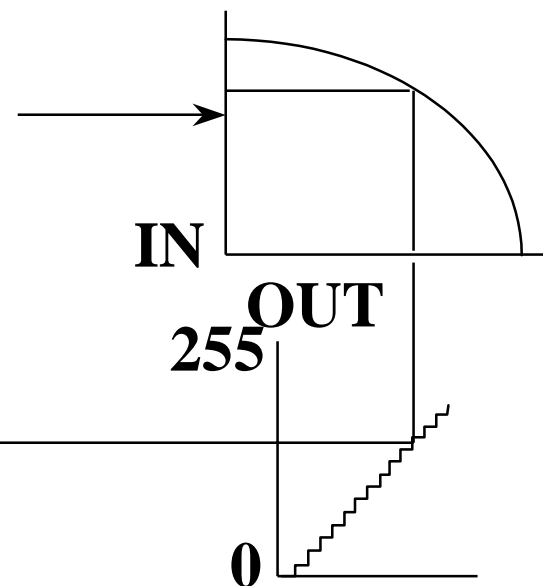
Sampled Image

Single Pixel
Continuous Representation
of Brightness



Scalar Quantization

170



Need For Image Compression: Storage




- **Digital Camera:**
- **Hybrid Image Capture**
- **Computed Radiography**
- **Graphic Arts**
- **Full-Motion Video**

Significance



- **Digital video and imaging will often (if not always) include compression**
- **Streaming applications on the verge of wide distribution and availability**
- **An important enabling technology that ‘educated’ computer and engineering developers need to understand to leverage**
- **An evolving discipline**
- **Many opportunities for entrepreneurs**

Digital Image Compression Applications

- 
- Consumer Imaging
 - Multimedia Products
 - Medical Imaging Remote Sensing
 - Photojournalism
 - Graphic Arts
 - Computer-Generated Images
 - Facsimile Transmission
 - Broadcasting – Streaming video
 - Many more such as: image archiving, digital copiers, desktop and electronic publishing, teleconferencing, video phone, education, law enforcement, government, etc.

Need for Image Compression: Transmission



- **Transmittal of imagery has driven the development of standard methods.**
 - Sales of compression equipment needs standards to insure interoperability.
- **Application Areas:**
 - Photojournalism
 - Facsimile Transmission
 - Medical Imaging

Need for Image Compression: Transmission



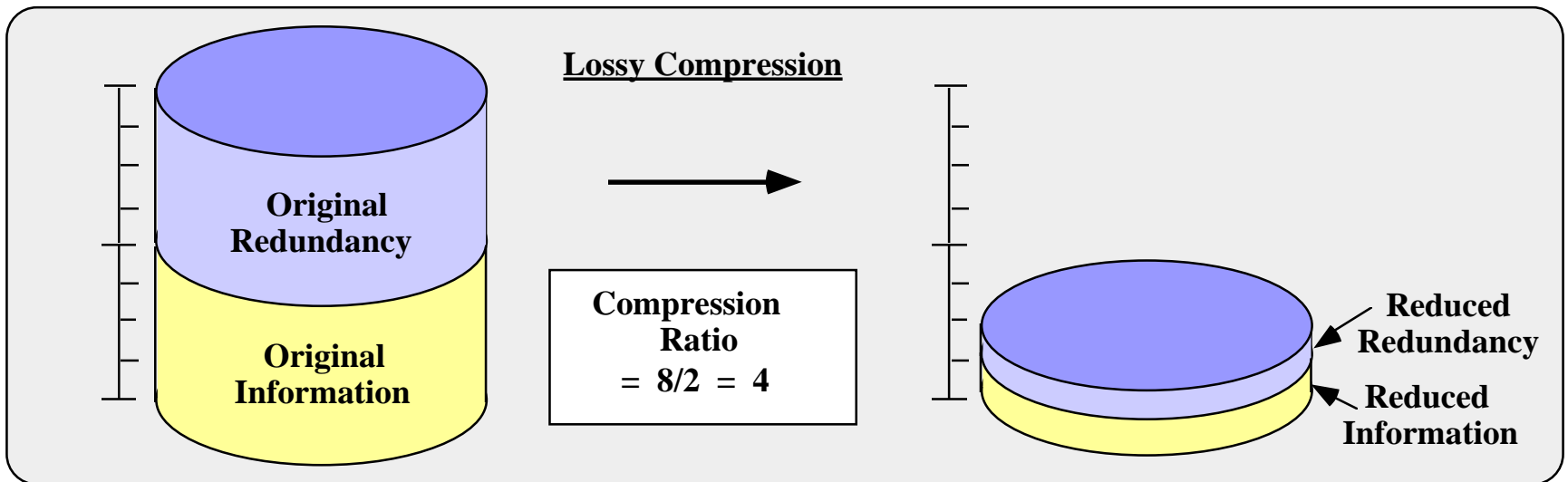
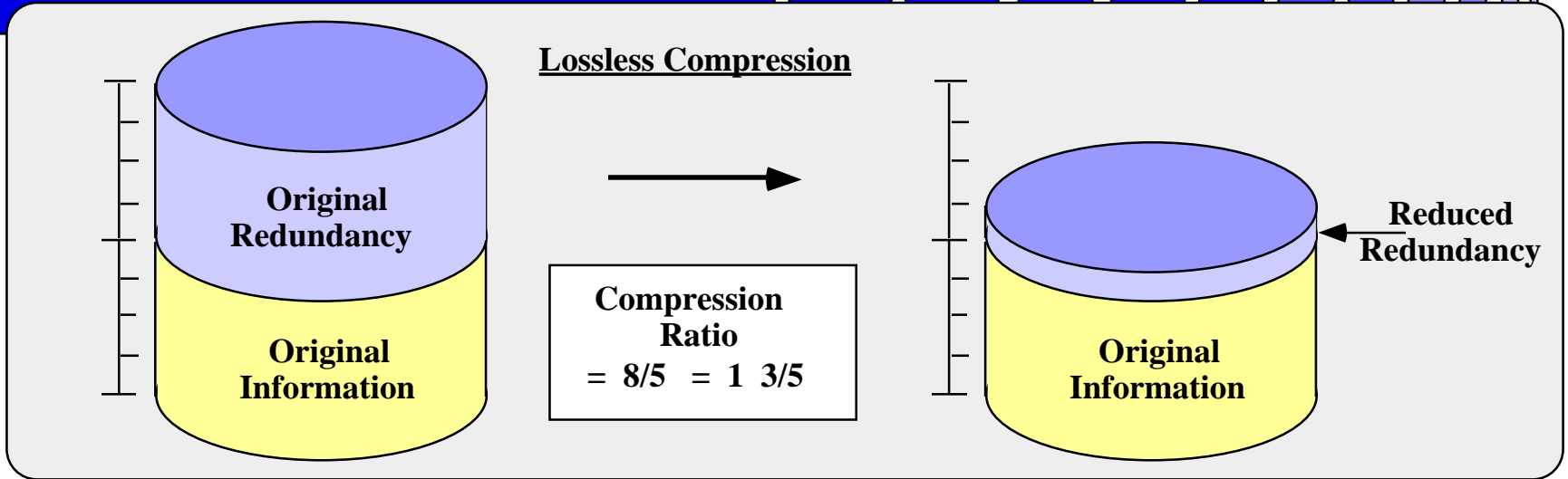
- **Video information levies a higher requirement on compression:**
 - Temporal information can be exploited for redundancy.
- **Application Areas:**
 - Videoconferencing and Videophone
 - Digital Broadcasting
 - Interactive Games (internet)

Why Can Images Be Compressed?




- **Image compression can be achieved primarily because image data are highly redundant. The degree of redundancy determines how much compression can be achieved. Three types of redundancy can be identified:**
 - **Spatial Redundancy**
 - **Spectral Redundancy**
 - **Temporal Redundancy**
 - **Psycho-visual Redundancy**


Lossless and Lossy Compression




Digital Image Compression Standardization Activities

- 
- **The adoption of image compression standards allows for:**
 - The exchange of images across application boundaries as well as a means for controlling image quality.
 - A significant reduction in the cost of specialized hardware required in real-time image compression systems, thus stimulating product development.
 - **Current standards:**
 - JPEG - Still Image Compression Standard
 - JBIG - Bilevel Compression (Not used much)
 - MPEG -1, MPEG -2 - Video Compression Standards for CD-ROM's, internet and television.
 - H.261 and H.263 - Interactive Video Standards.


Future Standards

- 
- **New Standards Development in Still Image and Video Compression**
 - **JPEG -2000 - Possibly a Wavelet Based method - Capable of up to 300:1 ratio of compression**
 - » **Much discussion in news group**
 - **New JBIG (Joint Bi-Level Image Processing Group) Compression Standard (Arithmetic Coding) – nicely described in Sayood**
 - **MPEG - 4 - Low Bit Rate Video Compression**
 - » **Microsoft**
 - **MPEG - 7 (follow-on to MPEG-4) Content searchable video**

Introduction To Image Processing

- 
- **What Does An Image Processing, Analysis, Understanding System Consist Of?**
 - **What Do We Need To Know To Specify, Evaluate Or Design A System**
 - Software & Algorithms
 - Data Structures & Data Bases
 - Processor Architectures
 - Available Commercial Equipment
 - **Applications Examples**

Key System Elements

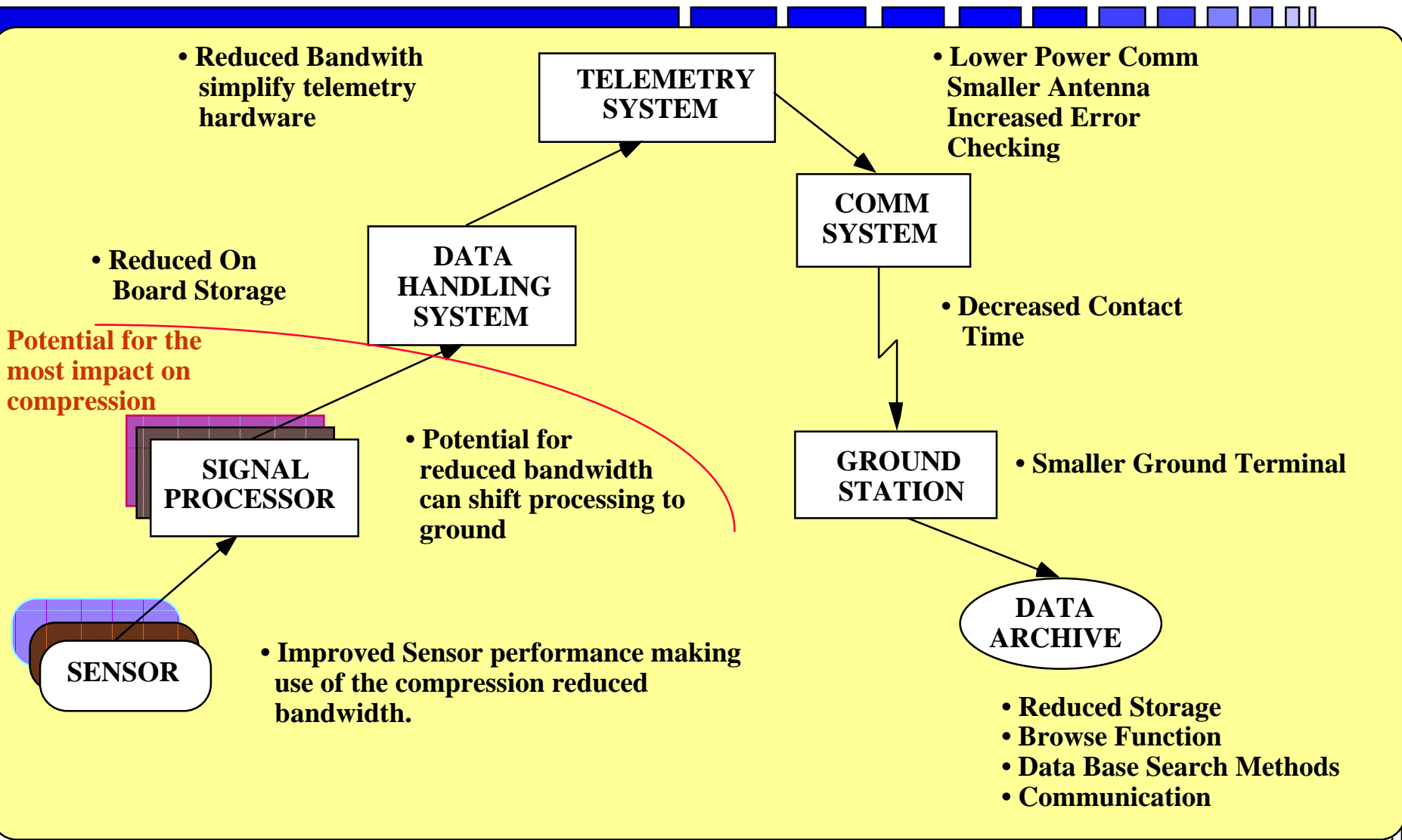
- 
- **Image Input**
 - Imaging Systems, Imaging Sensors, Image Sampling & Digitization
 - Smart Sensors
 - **Image Output**
 - Display Systems, Image Reconstruction
 - **Processors**
 - General Purpose Computers, Special High-Speed Processors
 - LSI/VLSI/CCD Processors
 - **Software & Algorithms**
 - Major Software Components, Languages, Libraries,
 - Design Principles
 - **Data Bases & Data Structures**
 - Structuring Data For Efficient Processing, Storage & Retrieval

Data Compression - Systems Issues

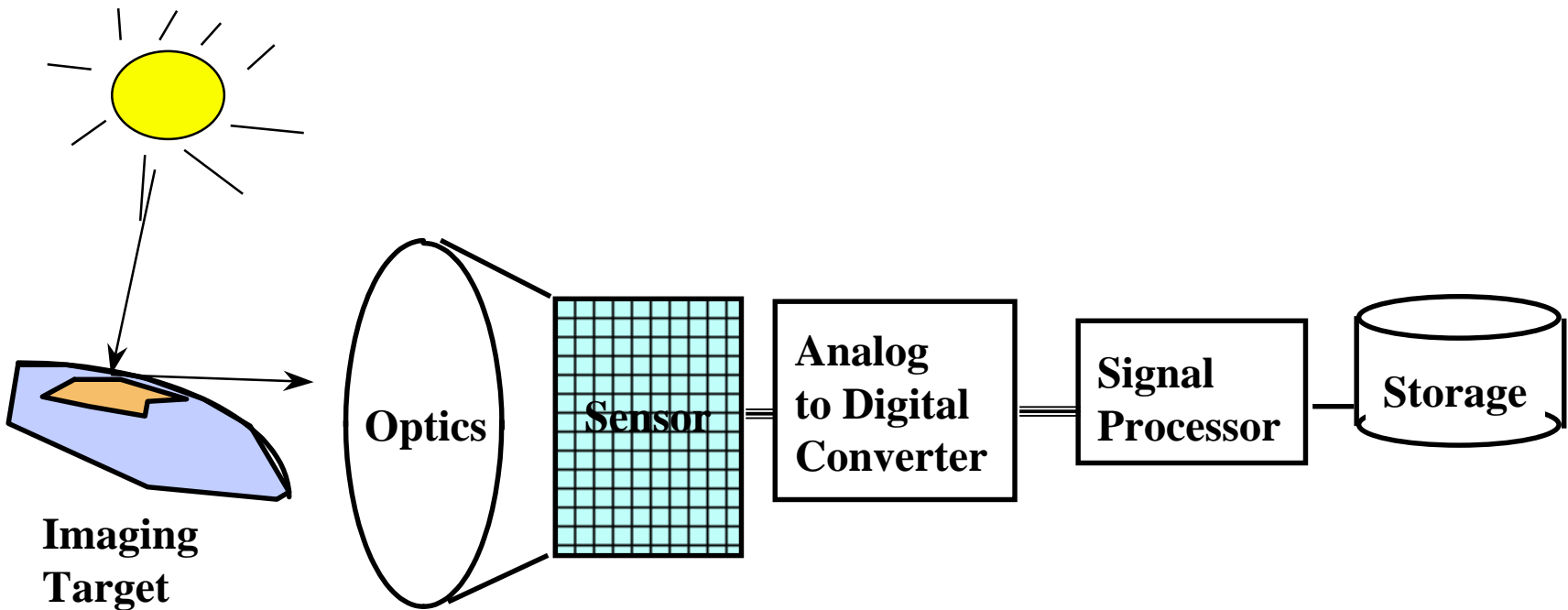


- **The design of a data compression system addresses many systems issues:**
 - **What fidelity of data is required by the mission**
 - **What precision is required by:**
 - » **Data Generation Process**
 - » **Data Collection Process**
 - » **Data Processing**
 - » **Data Interpretation**
 - **Is performance measured by absolute standards or in-exact human interpretation.**

System Applications of Data Compression



Data Collection/Compression Requirements



**Image
Formulation
Geometry**

**Optical
Sensor
Trades
Requirements**

**Electronics
Signal Processing
Trades
Requirements**

Image Formulation



- **Image is formed by the interaction of light on the target**
- **Image is detected by the reception of light from the target**
 - Target may reflect light
 - Target may emit light
- **Optical System will form the image on the detector**
 - Optical System will distort the image
 - Detector will also distort the image
- **Signal Processing hardware will introduce noise in the process**
 - Sample noise
 - Quantization Noise

Image Formulation


- 
- **Models exist that describe the effect of light passing through an atmosphere prior to hitting the target**
 - **Materials models will describe the effect of light reflecting on the target**
 - **Atmospheric models describe the effect of light after it hits the target prior to passing through the optical system**
 - **Optical System Equations describe the effect of the lens on the source.**

Image Formation



- A simple Example: The Pinhole camera

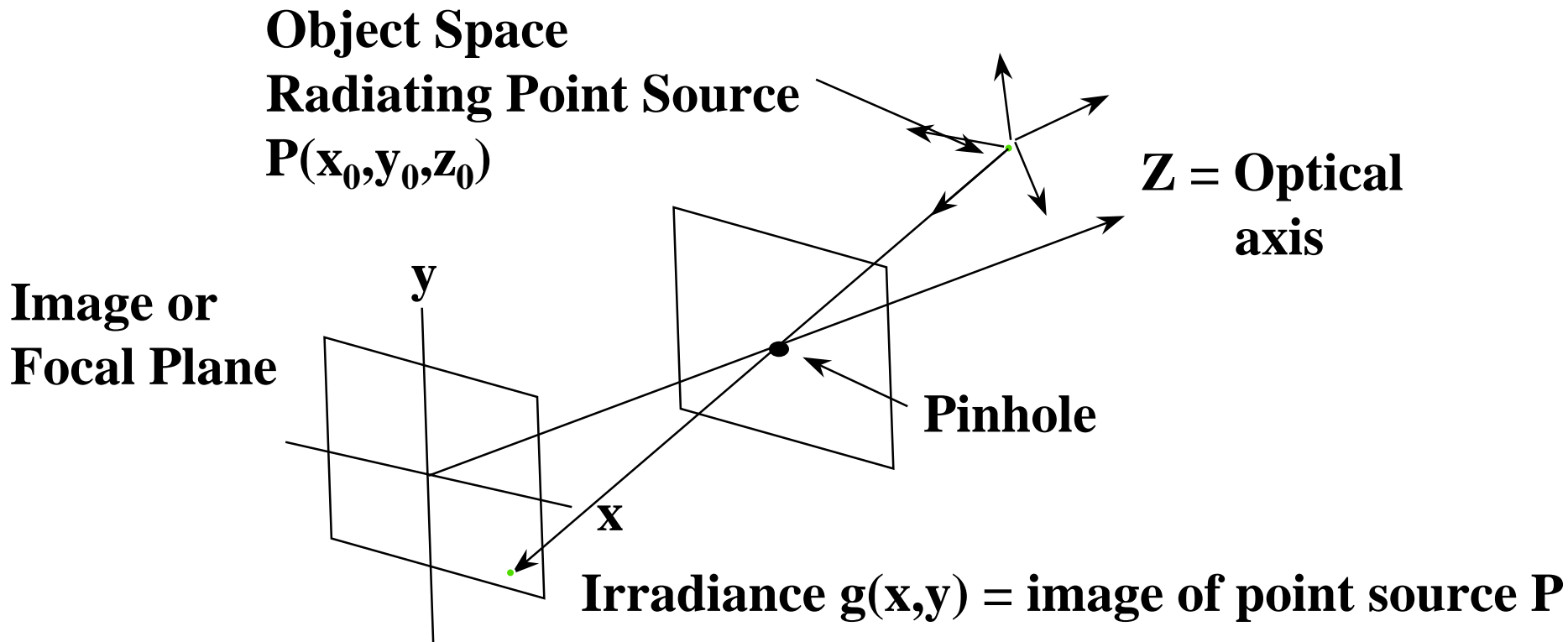
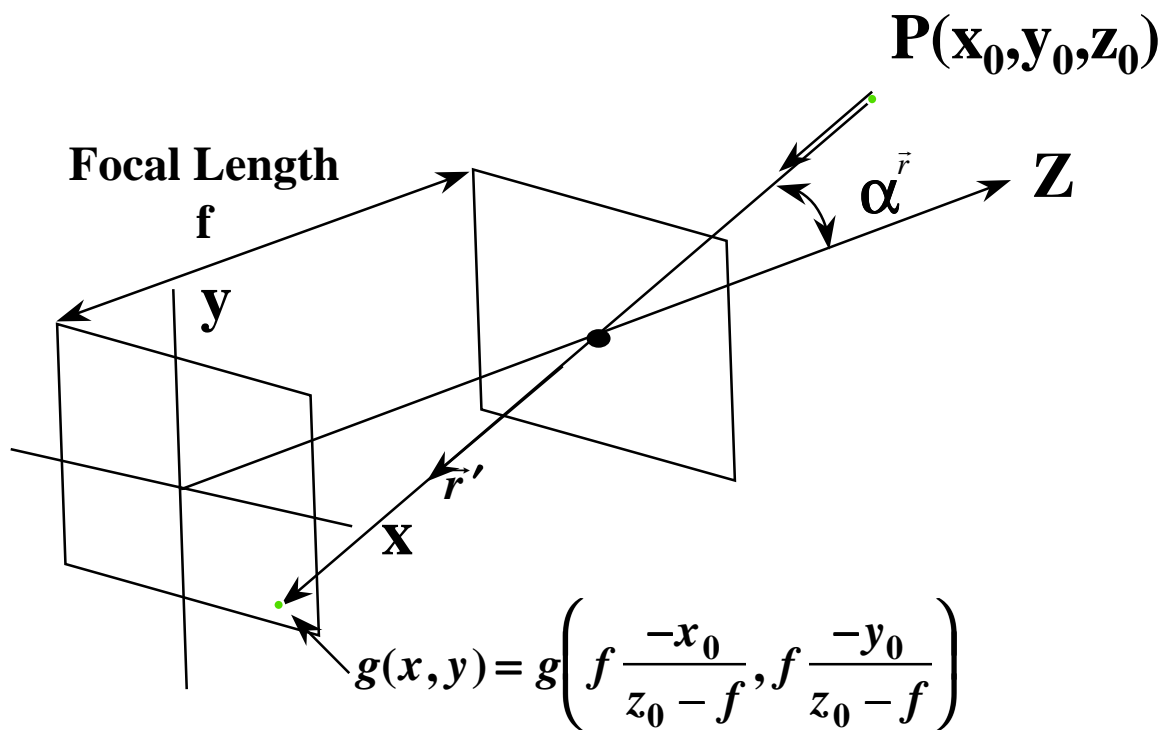


Image Formation



(f designates focal length here)

$$(Z_0 - f) = \bar{r} \cos \alpha$$

or

$$\bar{r} = (Z_0 - f) \sec \alpha$$

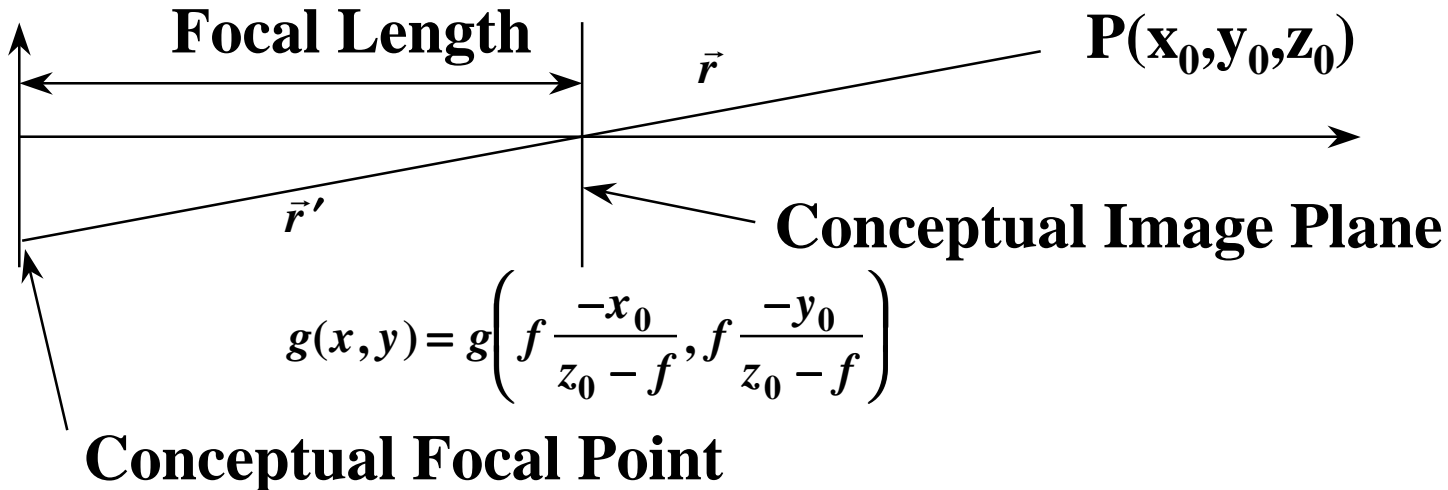
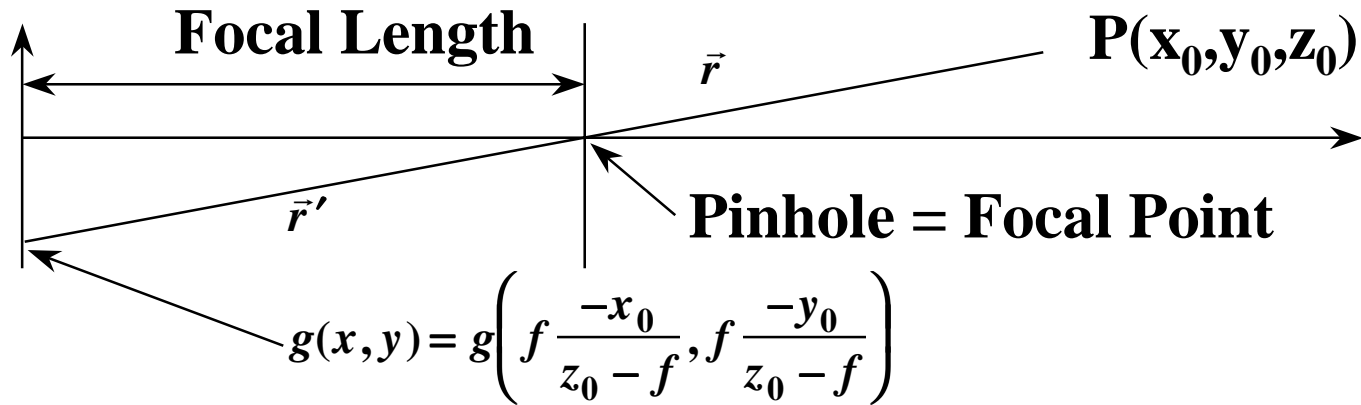
$$\bar{r}' = f \sec \alpha$$

so

$$\frac{\bar{r}'}{\bar{r}} = \frac{f}{(Z_0 - f)}$$

$$\bar{r}' = f \frac{\bar{r}}{(Z_0 - f)}$$

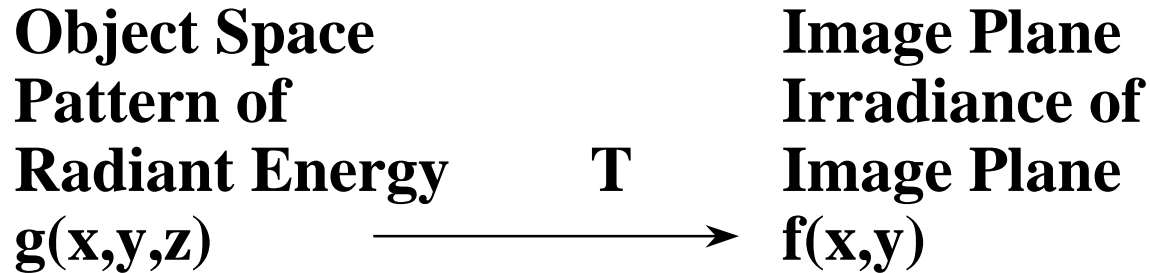
Focal Length Equations



Imaging System



- **Imaging System as a Mapping or Transformation**



- **Mapping or transformation is almost always linear or a linear sum of point sources**

$$T[ag_1(x,y) + bg_2(x,y)] = aT[g_1(x,y)] + bT[g_2(x,y)]$$

$a, b =$ any constant

$g_1, g_2 =$ any two functions

Image Formulation



- **Mathematical expression for unit point source P at coordinate (ξ , η)**

$$\delta(x - \xi, y - \eta) = \begin{cases} \infty & : x = \xi, y = \eta \\ 0 & \text{otherwise} \end{cases} \text{ Called a Dirac Delta Function}$$

with $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(x, y) dx dy = 1$

2 – dimensional delta function $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x, y) \delta(x, y) dx dy = g(0,0)$

Image Formulation



- Permits to write:

$$g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(\xi, \mu) \delta(x - \xi, y - \eta) d\xi d\eta$$

SIFTING INTEGRAL

Linear sum of point sources $\delta(x - \xi, y - \eta)$ $\begin{matrix} = 1 \text{ if } x = \xi \\ y = \eta \\ = 0 \text{ otherwise} \end{matrix}$

Image Formulation

- **Substitute in**

$$f(x, y) = T[g(x, y)]$$

$$\begin{aligned} f(x, y) &= T \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(\xi, \eta) \delta(x - \xi, y - \eta) d\xi d\eta \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(\xi, \eta) T \delta(x - \xi, y - \eta) d\xi d\eta \end{aligned}$$

Image Formulation

- **System Impulse Response:**

$$h(x, \xi, y, \eta) = T\delta(x - \xi, y - \eta)$$

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(x, y) h(x, y) dx dy = h(0, 0)$$

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(x - \xi, y - \eta) h(\xi, \eta) d\xi d\eta = h(x, y)$$

(Image of point source at coordinate ξ, η)

General case : Four variables (Space Variant)

Image Formulation

- **Space invariant impulse response:**

- If a point source yields the same image regardless of coordinate (ξ, η) then

$$h(x, \xi, y, \eta) = h(x - \xi, y - \eta)$$

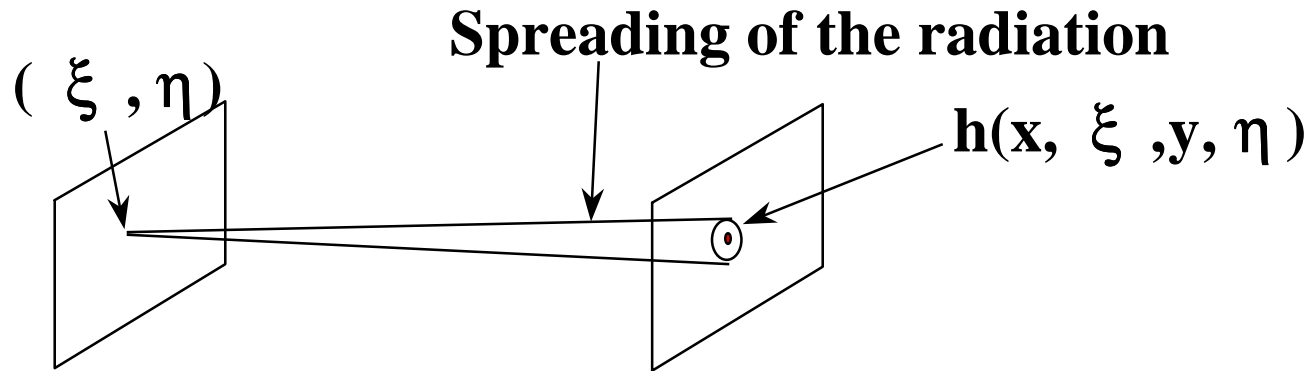
$$\text{and } f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(\xi, \eta) h(x - \xi, y - \eta) d\xi d\eta$$

Convolution

$$\text{Notation : } f(x, y) = g(x, y) \otimes h(x, y)$$

- If system is space-invariant: properties of imaging system can also be expressed in the spatial frequency domain.

Image Formulation



- An impulse source at $x = \xi$, $y = \eta$ results in a diffuse blur having an impulse response $h(x, \xi, y, \eta)$
- Knowing this response for all input points it will enable us to calculate the output $f(x, y)$

Uniform Sampling



(512,512)



(256,256)



(128,128)



(64,64)



(32,32)



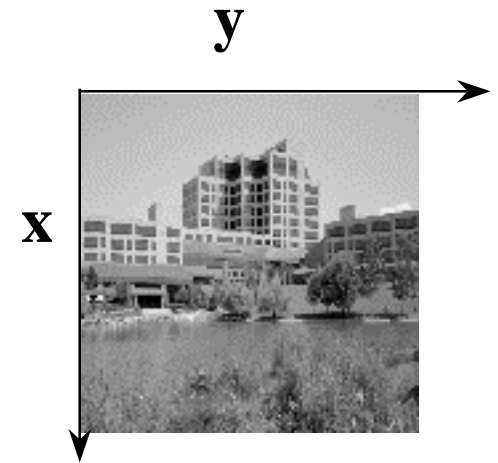
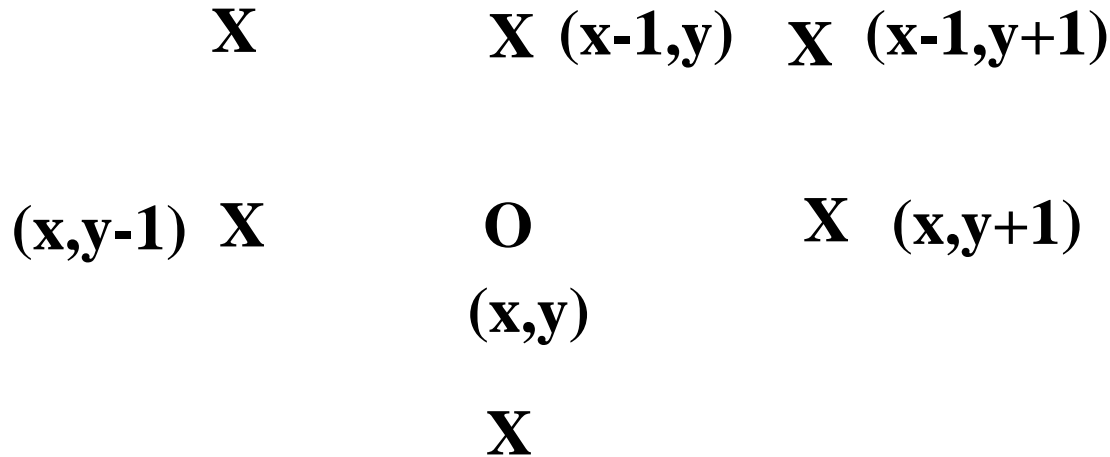
(16,16)



Image Orientation



- An image consists of an array of pixels, usually (512x512) (can be much higher)
- Neighbors of a pixel:



Number of Gray Levels



256



64

Eye can
detect 6 bits
(64 levels)



128



32

Number of Gray Levels



16



4



8




2

General Principals of Data.



- **Study of the characteristics of the data will point to the way to compress the data**
- **Effective compression relies on de-correlating the data.**
- **Correlation of Data**
 - **Spatial Redundancy - Correlation between adjacent data points**
 - **Spectral Redundancy - Correlation between different color planes or sensors**
 - **Temporal Redundancy - Correlation between different frames in an image**

Basic Terminology

- 
- **Data to be compressed is referred as:**
 - String, file text or input
 - **Data is generated by a source.**
 - **Source supplies the compressor with symbols over some alphabet.**
 - **Input symbols may be bits, characters, words, pixels, gray levels, voltage levels, etc**
 - **Compression is sometimes called source coding.**
 - **Compression Ratio is the ratio of the size of the original data to the size of the compressed data.**
 - **We also use the number of bits per symbol as an efficiency measures.**

Basic Methods of Compression



- **Statistical**
 - Each symbol is assigned a code based on the probabilities that it will occur
 - Highly probable symbols get short codes and vice versa
 - » Arithmetic Coding
- **Dictionary**
 - Groups of consecutive characters or phrases are replaced by a code
 - the phrase represented by the code can be found by looking it up in some dictionary.
 - » Ziv-Lempel Coding

Domain Principle



- **Statistics of Data**

- **Data Types**

- » **Data is a null entity.**


- Data has meaning only to an application
 - Applications will dictate whether a compression method should be lossless or lossy

- » **Data domain has specific meaning. (Sound, sensor, data set, text, etc)**

- » **Data characteristics:**

- Size (number of bits, bytes, etc)
 - Variability
 - Dynamic Range
 - Local versus Global Statistics

Data Characteristics

- 
- **Problem is in identifying the correlation, and devising a method to remove it.**
 - **Data has a statistical reach ? (Dynamic Range, statistical behavior)**
 - **Statistical description of data**
 - Average
 - Max-Min
 - Mean
 - Std Deviation
 - Skew
 - Kurtosis

Information Theoretic Measures



- **Shannon Information Theory**
- **Entropy Measures**
- **Data has characteristics such as volume (contained in some fixed length space)**
 - Data has redundancy part
 - Data has an information part
- **When compression reduces the redundant part only, the data can be reconstructed exactly (reversible)**
- **When compression reduces the redundant and information part, the data can not be reconstructed exactly (irreversible)**

Lossless Compression



- **Can recover or reconstruct image/data exactly (no degradation)**
- **Compression achieved is image/data dependent**
- **Usually yields modest compression (~ Factors of 2 - 3)**
- **Usually generates variable length compressed output**
- **Most effective on textual data**
 - Text
 - Programming language source or object code
 - Binary data

Lossy Compression



- **Reconstructed image/data is an approximation of the original**
- **System Tradeoff: Compression vs Degradation**
- **Larger achievable compression (Factors >3)**
- **Generates fixed length and Variable length compressed output**
- **Techniques are usually designed with fixed compression and variable (data dependent) degradation or constant quality, with variable compression.**

Data Entropy

- **Defined as the expected value of self-information**
- **Measures the average amount of unique information in the data**
- **Measures the average a priori uncertainty in the data**
- **Gives an upper bound on the compression achievable via a lossless pixel compression technique**

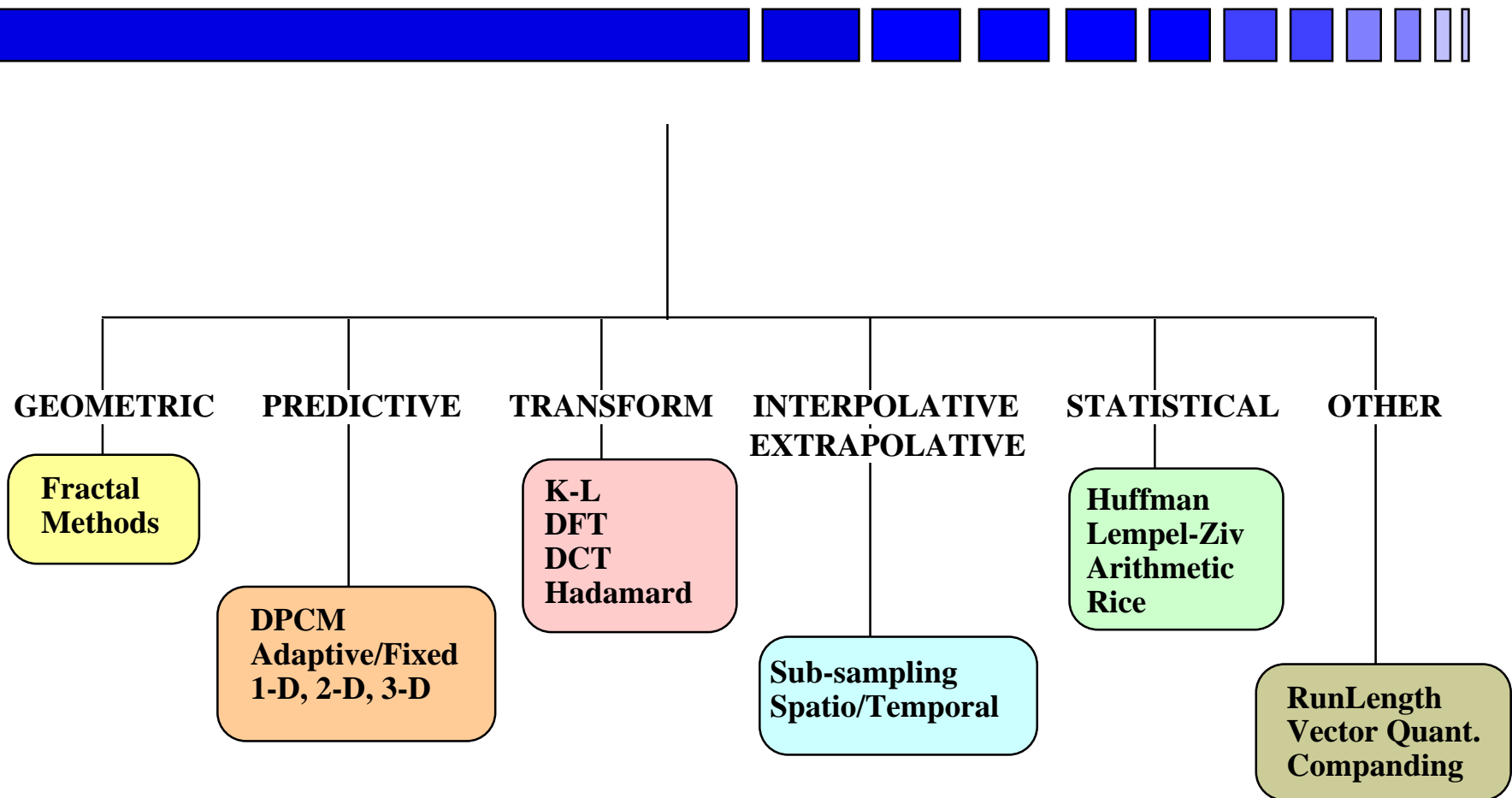
$$H = - \sum_{j=0}^{N-1} \Pr(x_j) \log_2 (\Pr(x_j)) \quad \text{bits}$$

N = Number of quantization levels

x_j = j^{th} quantization value

$\Pr(x_j)$ = Probability of occurrence of the j^{th} quantization value

Data Compression Techniques



Timeline of Compression Techniques:




- **Huffman - 1952**
- **Transform Based Compression (1970's)**
- **Ziv-Lempel Compression (Ziv and Lempel, 1977)**
- **Rice Coding (Rice 1979)**
- **Arithmetic Coding (Guazzo, 1980, Langdon 1984)**
- **Fractal Compression (1990)**
- **Wavelet Compression (1990's)**
- **Second Generation Image Compression (1993)**
- **Model Based Compression (MPEG-4) 1997**

Information Theory Concepts



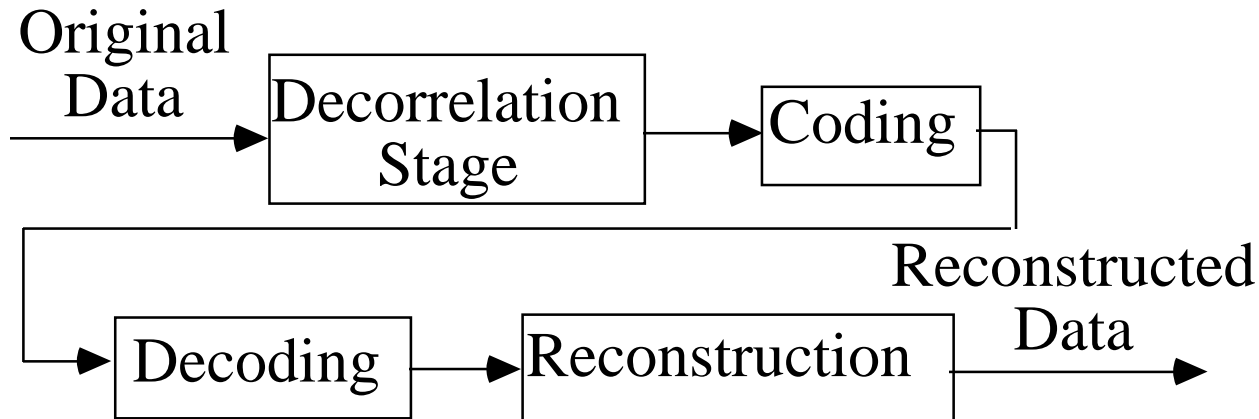
- **Source Models**
- **Compression Systems assume some underlying model for data generation**
- **Understanding the model will aid in the development of a efficient code**
 - Code based on a “True” Model will have greater performance
 - Performance is measured as approaching the entropy of the data

Data Models


- 
- **Data Compression methods always assume some model for the data.**
 - **Model for data generation.**
 - **The data product will produce similar statistics.**
 - **Markov Models**
 - » **Good for Text generation systems**
 - » **Statistical Basis**
 - **Laplacian Models**
 - » **Image Generation Model**
 - » **Image statistics seem to be Laplacian**
 - **DPCM Image data**

Data Models

- Lossless methods may be statistically based.
- Coding is the second part of the compression problem. Decorrelation is the First part



Decorrelation Consists of:

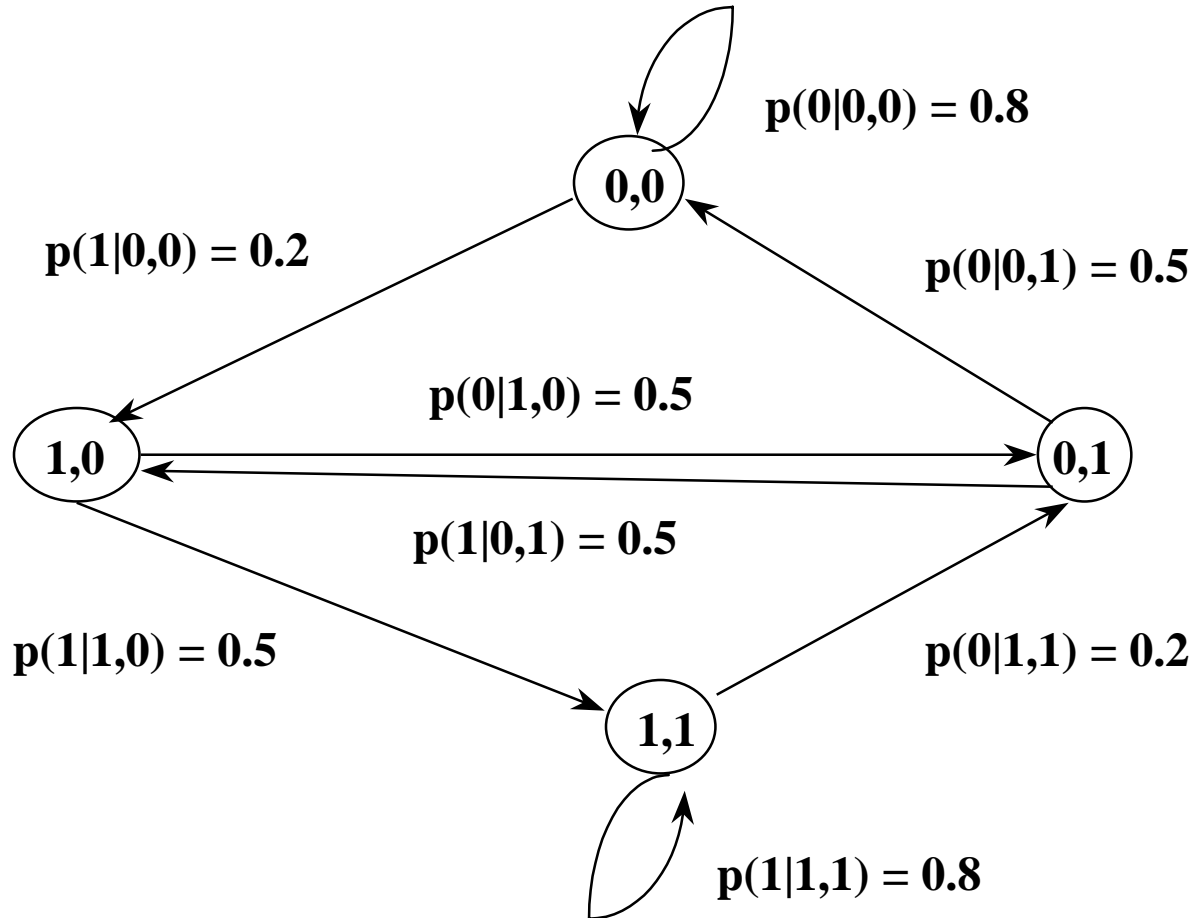
- 
- A horizontal bar composed of 12 rectangular segments. The first segment is a solid dark blue rectangle. The next seven segments are smaller, solid blue rectangles of decreasing width from left to right. The final four segments are the smallest, light blue rectangles, also of decreasing width from left to right.
- **Analysis of the Statistics of the raw Data**
 - **Transformation of the data into a new domain**
 - Analysis of the statistics of the new domain
 - **Compression has also been described as a two stage process, encoder and modeler**
 - **The modeler assigns probabilities to the symbols, and the encoder translates the probabilities to a sequence of bits.**
 - **Relationship between probabilities and codes were first described by Shannon (1948)**

Source Coding Theorem:




- A symbol that is expected to occur with probability p is best represented by $-p \log_2 p$ bits
 - A symbol with a high probability is coded with a few bits, and a symbol with a low probability requires many bits.
- The expected length of code can be found by averaging over all possible symbols, giving the formula:
 - Entropy = $-\sum p_i \log_2 p_i$
 - Measure of the amount of order (or disorder) in the symbols

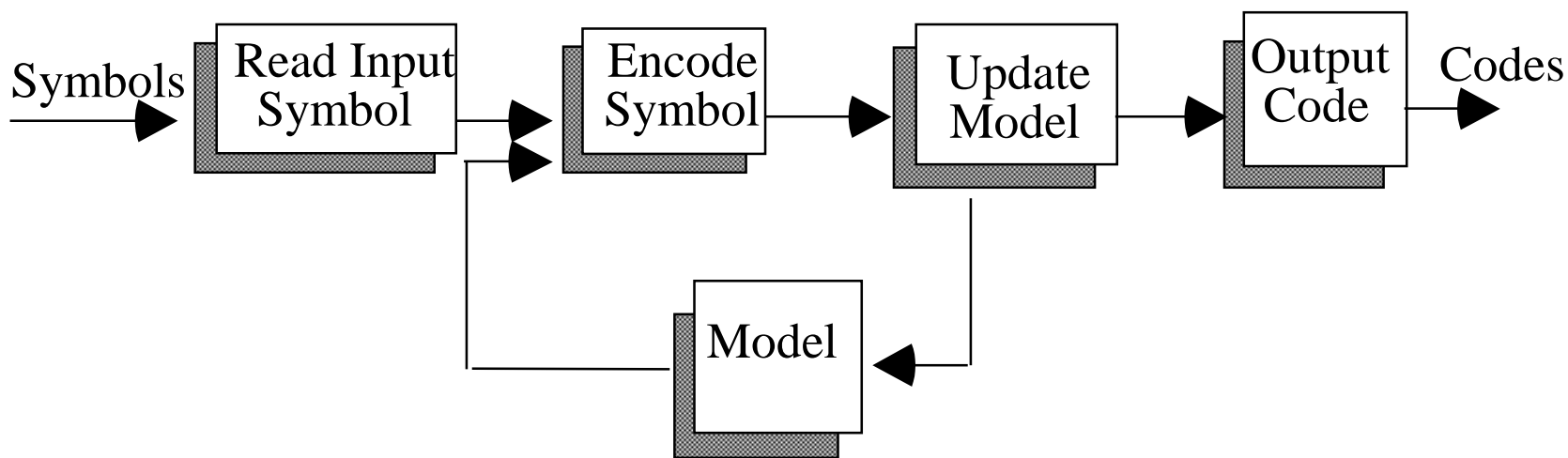
Markov Process State Diagram



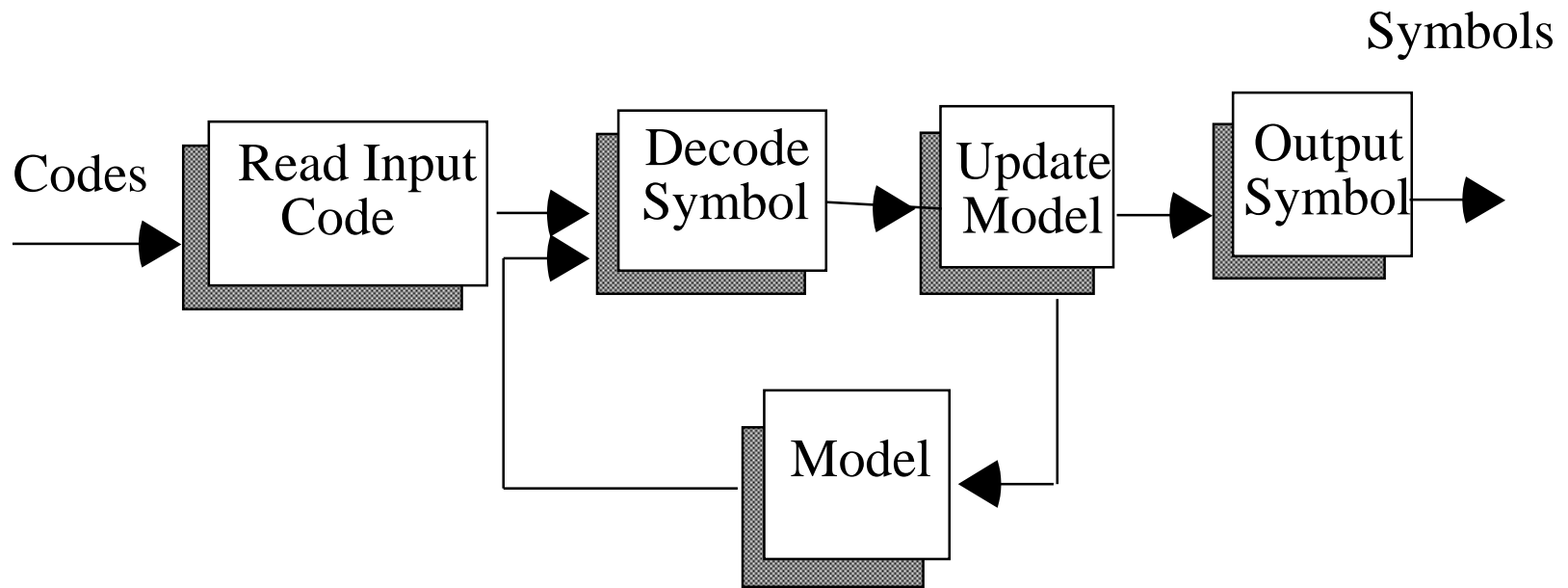
Role of Modeling

- 
- **Estimate a probability for each symbol**
 - From the probabilities, the entropy can be calculated
 - Entropy is a property of the model
 - » Given a model, the estimated probability of a symbol is sometimes called the code space allocated to the symbol.
 - » The interval between 0 and 1 is being divided up among the symbols, and the greater a symbol's probability, the more of this space it is taking from other symbols.
 - The best average code length is obtained from models in which the probability estimates are as accurate as possible.
 - » The accuracy of the estimate depends on how much contextual knowledge is used.
 - The model is essentially a collection of probability distributions, one for each context in which the character might be encoded.
 - Contexts are termed conditioning classes (they condition the probability estimates)

General Adaptive Compression



General Adaptive Decompression




Adaptive and Non-adaptive Models




- **Design of the Model will reflect the basic statistics of the image**
- **Three types of Models:**
 - **Static Models**
 - **Semiadaptive Models**
 - **Adaptive Models**


Static Models:

- 
- **Use the same model for all text. The model does not change as the text changes.**
 - **Identical model used in decoder**
 - **Drawback: scheme will give unboundedly poor compression when the text being coded does not correspond to the model.**
 - **Advantages: Speed and simplicity**


Semiadaptive Models

- 
- **Uses a different model for each text encoded**
 - **Before performing compression, the text (or sample) is scanned, and a model is constructed from it.**
 - **The model must be transmitted to the decoder before the compressed text is sent.**
 - **Extra cost in transmitting the model**
 - **Model is usually well suited to the text**


Adaptive (or dynamic) modeling

- 
- **Initially, both the encoder and decoder assume some bland model, such as all characters being equiprobable.**
 - **The encoder using this model to send the first symbol, and the decoder receives the symbol**
 - **Both the encoder and decoder update their models in the same way.**
 - **The next symbol is encoded and decoded using the new models.**
 - **The models are updated continuously, and will soon be well suited to the text.**

Models and Coding

- 
- **Data that approaches that ideal model will encode to a greater "efficiency"**
 - Efficiency can be smallest number of bits, or code to close to the information theoretic rate
 - Performance of an algorithm will change as the data changes.
 - » Typical Data Sets
 - » Generic data characteristics
 - » Mission specific data sets

Statistical Methods


- 
- **Huffman Codes**
 - **Rice Codes**
 - **Fractal Compression**
 - **Vector Quantization**
 - **Why Fractal Compression and Vector Quantization?**
 - **Fractal Compression** relies on a **Statistical** measure of closeness to a geometric model
 - **Vector Quantization** relies on a centroid measure to pick the appropriate vector

The Shannon-Fano Algorithm



- **Designed to define an effective Code table**
- **For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.**
- **Sort the list of symbols according to frequency, with the most frequently occurring symbols at the top and the least common at the bottom**

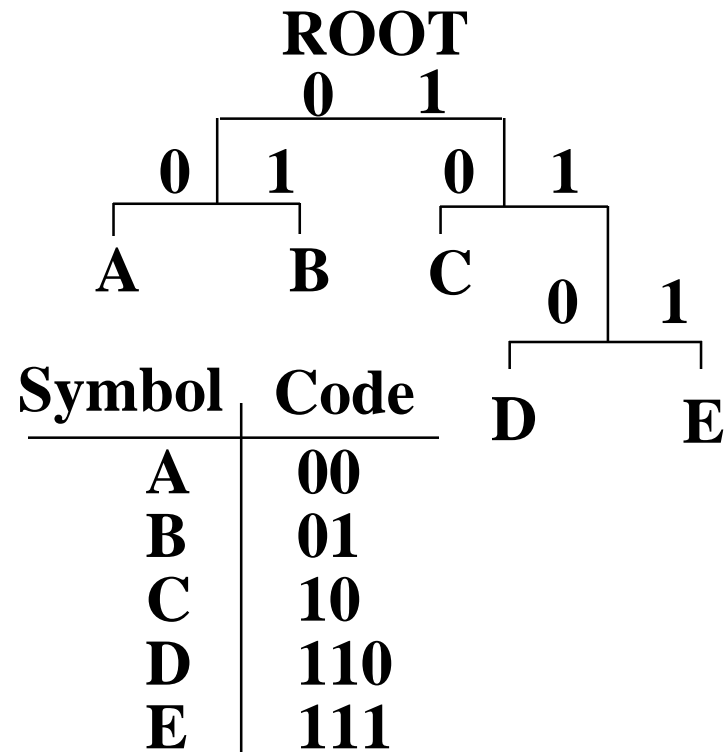
The Shannon-Fano Algorithm (Continued)

- 
- **Divide the list into two parts, with the total frequency counts of the upper half being as close to the total of the bottom half as possible.**
 - **The upper half of the list is assigned the binary digit 0, and the lower half is assigned the digit 1. This means that the codes for the symbols in the first half will all start with 0, and the codes in the second half will all start with 1.**
 - **Recursively apply the same procedure to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.**

The Shannon-Fano Algorithm



Symbol	Count
A	15
B	7
C	6
D	6
E	5



The Shannon-Fano Algorithm



- Put the dividing line between symbols B and C assigns a count of 22 to upper and 17 to lower (close to half)
- A and B will have a code that starts with 0 and C, D, and E will have a code starting with 1

Symbol	Count		
A	15	0	Upper Group
B	7	0	22
First Division			
C	6	1	Lower Group
D	6	1	17
E	5	1	

The Shannon-Fano Algorithm



- Upper half gets a new division between A and B which puts A on a leaf with code 00 and B on a leaf with code 01.
- In final table, three symbols with the highest frequency have 2 bit codes, two symbols with lower frequencies have been assigned 3-bit codes.

Symbol	Count				
A	15	0	0		
		Second Division			
B	7	0	1		
		First Division			
C	6	1	0		
		Third Division			
D	6	1	1	0	
		Fourth Division			
E	5	1	1	1	

The Shannon-Fano Algorithm

- *Information Content* = $\log_2(\text{symbols_probability})$
- *Information Bits* = *Count* * *Information Content*
- The information adds up to 85.25 bits. The ASCII storage requirements is 39*8 bits (312 bits). The Shannon-Fano representation is 89 bits.


Symbol	Count	Info Cont.	Info Bits	SF Size	SF Bits
A	15	1.38	20.68	2	30
B	7	2.48	17.35	2	14
C	6	2.70	16.20	2	12
D	6	2.70	16.20	3	18
E	5	2.96	14.82	3	15

The Huffman Algorithm

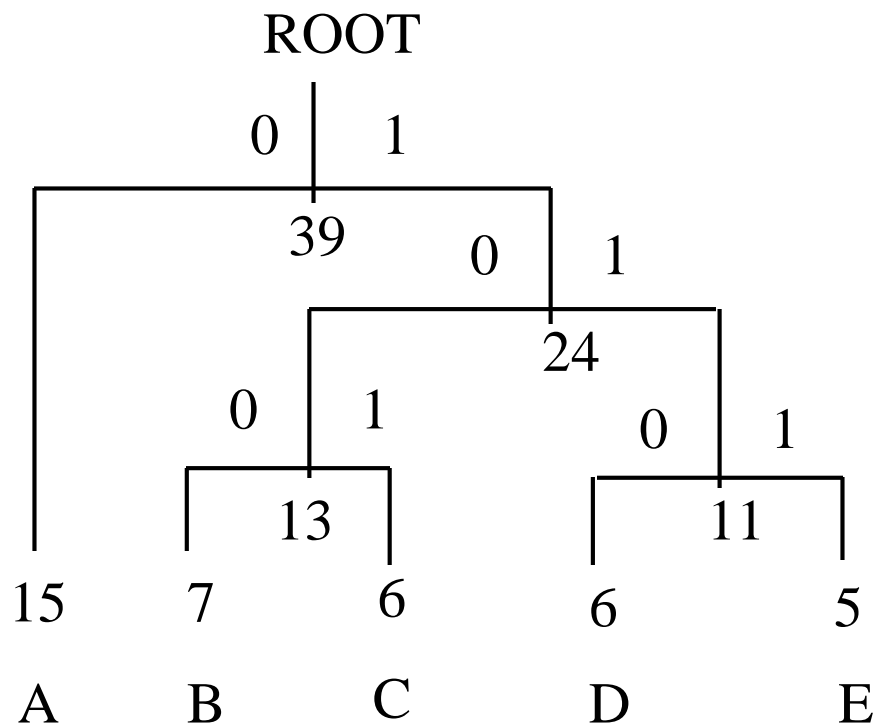


- **Huffman coding shares most characteristics of Shannon-Fano Coding.**
 - Variable Length Codes - integral number of bits
 - Symbols with higher probabilities get shorter codes
 - Huffman codes have the unique prefix attribute - No code is a prefix to another code
 - Shannon-Fano codes are developed top down - assign the most significant bits to each code, working down until the tree is complete
 - Huffman codes are developed bottom up, starting with the leaves of the tree, and working closer to the root
- **Individual symbols are laid out as a string of leaf nodes connecting a binary tree.**
- **Each nodes weight is the frequency of the symbol**

Procedure for Huffman code development

- 
- Two free nodes with the lowest weights are located
 - A parent node for these two nodes is created. It is assigned a weight equal to the sum of the two child nodes
 - The parent node is added to the list of free nodes, and the two child nodes are removed from the list
 - One of the child nodes is designated as the path taken from the parent node when decoding a 0 bit. The other is arbitrarily set to the 1 bit.
 - The previous steps are repeated until only one free node is left (the root of the tree)

Huffman Code



Symbol	Code
A	0
B	100
C	101
D	110
E	111

Comparison Between Shannon-Fano Codes and Huffman Codes

- Shannon-Fano Code is 89 bits, Huffman Code is 87 bits

Symbol	Count	Shannon-Fano Size	Shannon-Fano Bits	Huffman Size	Huffman Bits
A	15	2	30	1	15
B	7	2	14	3	21
C	6	2	12	3	18
D	6	3	18	3	18
E	5	3	15	3	15

Modified Huffman Codes




- **Most of the symbols in a large symbol set would have very small probabilities.**
- **These symbols would take a disproportionately large share of the codeword memory since the length is roughly proportional to its information content:**
 - **Information Content = $\log_2(\text{symbols_probability})$**
- **Advantageous to lump the less probable symbols into a symbol called ELSE, and design a code for the reduced set.**
 - **Whenever a symbol belonging to ELSE needs to be encoded, the system will transmit the code for ELSE followed by the actual value.**
- **Group 3 Fax uses this concept**
- **Coding of Quantized transform coefficients also use this approach since there are a large number of zero values**

Group 3 Fax




- Each binary scan line is regarded as a sequence of alternating black and white runs which are encoded with separate variable length code tables.
- **RUN** = The number of times a particular values occurs along the scan line
- Each run can occur over the scan line (up to 1728 pixels).
 - Rather than have a Huffman table with 1728 entries, the modified table has 0-63 entries, with the remainder encoded as $64N+M$, where N is an integer between 1 and 27.
 - M will be encoded using the table (since this is a modulo 64 code)
 - The next 27 entries of the Huffman table are codewords for the value of N .
 - The value of M for these runs is then encoded using the first 64 entries:
 - If $\text{run} = 213$, $213/64=3.3281$, therefore $N = 3$, $M=213-64*3=21$
 - The Huffman table code consist of entry 67 ($64+3$), followed by the entry 21 of the table.


Limitations of Huffman Coding

- 
- **Ideal binary codeword length for a source symbol s_i from a DMS is $-\log_2 p(s_i)$**
 - Codeword lengths must be integers, the condition is met only when the source symbol probabilities are negative powers of two (1/2, 1/4, 1/8, etc.)
 - If the symbol probabilities significantly deviate from this ideal situation, direct encoding of the individual source symbols can result in poor code efficiency


Limitations of Huffman Coding

- 
- **To improve the coding efficiency, encode the symbols of an extended source rather than the original source.**
 - Since the number of entries in the Huffman code table grows exponentially with the block size, it may be necessary to settle for an inefficient code to avoid excessive implementation complexity.
 - If the Huffman code is used to encode sequences generated by a Markov source, the conditional probability distribution of each symbol is different.
 - To code the source at a rate near its entropy, a separate Huffman table is needed for each state.
 - For the entropy per original source symbol of the adjoint source (extended source), to approach entropy of the Markov source, the block size must be large.

Limitations of Huffman Coding

- 
- **Huffman coding cannot efficiently adapt to changing source statistics**
 - Preset Huffman code with different sources may actually result in data expansion.
 - In order to provide adaptivity, Huffman coding is often implemented as a two pass process.
 - » First pass gathers the symbol statistics and generates the codebook, and the second pass encodes the data
 - Approach works well when the symbol probabilities are constant within a given image
 - In many situations, the symbol probabilities change throughout an image, and the use of a fixed Huffman code can severely affect the coding performance.

Arithmetic Coding

- 
- In order to achieve a reasonable coding efficiency with Huffman coding, the sequence generated by the source is divided into blocks and each block is assigned a codeword.
 - At the decoder, the received sequence is parsed into variable-length blocks corresponding to individual codewords.
 - Arithmetic code is a non-block code where a codeword is assigned to the entire input sequence s_m of length m symbols.
 - Slightly different source sequences can result in significantly different code sequences.
 - Codeword length approximately equals $-\log_2 p(s_m)$ where $p(s_m)$ is the probability of the source sequence s_m

Arithmetic Coding (Elias Code)

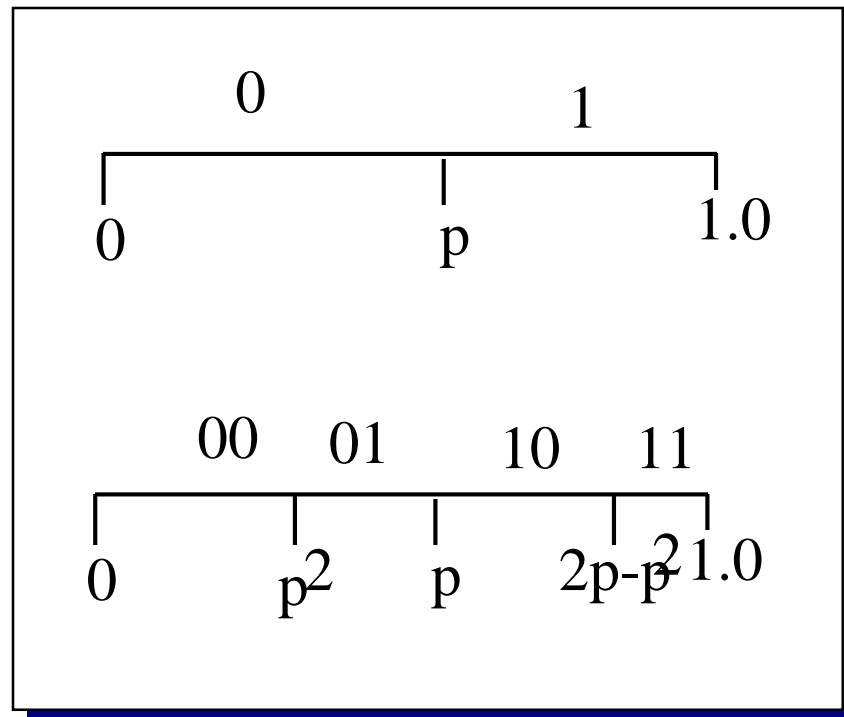


- Encoding a sequence of binary symbols, s_m generated by a DMS.
- Let p denote the probability of occurrence of a '0' and $q = 1-p$ denote the probability of occurrence of a '1' in this sequence.
- Denote I by the half-open interval $[0,1)$
- Since the sum of $p(s_m)$ over all the 2^m possible source sequences of length m is one, it is possible to assign a subinterval $I_l, l=1,2,\dots,2^m$, within I , to each source sequence s_m , such that the length of I_l is equal to $p(s_m)$ and the subintervals are non overlapping.
- The subinterval assignment is accomplished as follows:
- The interval I is first partitioned into two subintervals $[0,p)$ and $[p,1)$.

Elias Code Partitioning



- The subinterval $[0,p)$ is chosen if the first symbol in the sequence is a '0', and the subinterval $[p,1)$ is chosen if it is a '1'.
- Then each subinterval is further partitioned into two subintervals in a similar fashion.
 - For a subinterval $[0,p)$, this partitioning results in two subintervals, $[0,p^2)$, $[p^2,p)$ and for the subinterval $[p,1)$, it results in $[p,2*p-p^2)$ and $[2*p-p^2,1)$



Elias Code Partitioning



- The subinterval $[0,p)$ is chosen if the first symbol in the sequence is a '0', and the subinterval $[p,1)$ is chosen if it is a '1'.
- Then each subinterval is further partitioned into two subintervals in a similar fashion.
 - For a subinterval $[0,p)$, this partitioning in results in two subintervals, $[0,p^2), [p^2,p)$ and for the subinterval $[p,1)$, it results in $[p,2p-p^2)$ and $[2p-p^2,p)$

Elias Code Partitioning




- **Together with the first symbol, the second symbol in the sequence is used to specify one of these subintervals.**
 - The source sequence '01' corresponds to the subinterval $[p^2, p)$.
 - Let us denote the subinterval specified after $j-1$ binary source symbols by $[L^{(j-1)}, R^{(j-1)})$, where L stands for the left endpoint and R the right endpoint.
 - The next symbol is a '0', a new subinterval $[L^{(j)}, R^{(j)})$ is specified where:
 - » $L^{(j)} = L^{(j-1)}$
 - » $R^{(j)} = L^{(j-1)} + p(R^{(j-1)} - L^{(j-1)})$,
 - and if it is a '1',
 - » $L^{(j)} = L^{(j-1)} + p(R^{(j-1)} - L^{(j-1)})$,
 - » $R^{(j)} = R^{(j-1)}$
 - For any sequence, it can be shown that the subinterval generated by this method has a width equal to the probability of the sequence.
 - The subintervals produced by all the possible sequences of length m are nonoverlapping, and their union completely covers the interval I .

Elias Code Partitioning



- Once the subinterval $[L_l, R_l)$, ($l=1,2,\dots,2^m$), corresponding to a certain source sequence s_m , has been identified, a codeword for can be constructed by expanding the subinterval beginning point, L_l , in binary form and retaining only the $n_l = \lceil -\log_2 p(s_m) \rceil$ bits after the decimal point.
 - By expanding the point in binary form, we mean for example, $0.8125 = 1/2 + 1/4 + 1/16 \rightarrow 0.1101$
 - The reason the codewords can be constructed in this manner is that the beginning point of each subinterval is separated from the beginning point of its nearest right-hand neighbor by the subinterval width $p(s_m)$, where $p(s_m) \geq 2^{-n_l}$
 - As a result, the first n_l bits of the binary expansions of the two adjacent subintervals cannot be identical, and only n_l bits are required to uniquely specify the subinterval I_l and hence to specify the source sequence s_m


Elias Code Partitioning

- 
- **The encoding operation has a sliding structure; as a few source symbols enter the encoder, a few code symbols leave the encoder.**
 - For example, as soon as the encoder has received enough source symbols to determine if I_l belongs to the interval $[0,0.5)$ or to $[0.5,1)$ it knows whether the first output bit is a '0' or a '1'.
 - Additional codeword bits are generated as I_l becomes more finely specified.
 - In a similar manner, the decoder can also start decoding after receiving only a few code symbols.
 - Note that there is no one-to-one correspondence between the codeword bits and blocks of input sequence bits as in the case in Huffman coding.

Elias Code Partitioning

- **The previous discussion was based on the assumption that the source is memoryless, but it can be extended to a Markov source.**
 - Consider the subinterval specified after $j-1$ source symbols.
 - The next partitioning is based on the value of p for the j^{th} symbol.
 - In the case of a Markov source, this value is not a constant and is determined by the previously encoded source symbols, (the state of the Markov source).
 - The previous equations can be extended to a Markov source by:
 - » $R^{(j)} = L^{(j-1)} + p^{(j)}(R^{(j-1)} - L^{(j-1)})$,
 - » $L^{(j)} = L^{(j-1)} + p^{(j)}(R^{(j-1)} - L^{(j-1)})$,
 - » where $p^{(j)}$ denotes the dependence of p on the position of the symbol within the source sequence.
 - » The decoder, as it decodes each symbol constructs the state corresponding to the next symbol, and uses the appropriate $p^{(j)}$ value to decode the next symbol.

Implementation Problems of Elias Code

- 
- **Precision required to carry out the subinterval computations**
 - As the length of the source sequence increases, the length of the subinterval specified by the sequence decreases, and more bits are required to precisely identify the subinterval.
 - Implementation of arithmetic coding uses a scaling (renormalization) strategy and a rounding strategy
 - » **Scaling:** magnifies each subinterval prior to partitioning so that its length is always close to 1
 - » **Rounding:** use a fixed bit length (b -bit) (finite precision) arithmetic to measure the subinterval and perform the partitioning
 - To be able to represent all the nonzero symbol probabilities, the value of b should be chosen such that the smallest symbol probability is always larger than 2^{-b}
 - Generally, a larger b results in less quantization and renders a more efficient code.

Other issues



- **Incremental transmission and reception.**
 - Encoding algorithm does not transmit until the final interval is determined.
 - As the interval narrows, the leading bits of the left and right end points become the same.
 - Any leading bits that are the same may be transmitted immediately, since they will not be affected by further narrowing.
- **Precision**
 - The precision required by arithmetic coding grows without bound as the length of the ensemble grows.
 - Fixed Precision registers may be used as long as underflow and overflow are detected and managed.

Other issues

- **Limits on Performance**

- Arithmetic coding will theoretically code to entropy, but the use of a termination character and non-infinite precision reduces the effectiveness.
- Since termination character is of very low probability, the impact should be low.
- Studies by Witten et al approximate the overhead due to the use of fixed precision at 10^{-4} bits per source message

IBM Q-Coder



- **Features:**
 - **Only encode Binary sequences**
 - » **Non-binary sources require a binary decision tree to characterize the source symbols.**
 - **Uses a 12 bit register to implement the coding operation but only uses 30 quantized values for the probability of the less probable symbol (LPS) (empirically determined)**
 - » **Approximations are made which allow the interval partitioning based on the probabilities to be approximated by subtraction or table look-up rather than multiplications**

IBM Q-Coder

- **Features:**

- Does not require an estimate of the probability of the symbols to be encoded; arrives at a robust probability estimate for a source symbol after encoding it only a few times.
 - » When encoding a binary Markov source, the value of q in each state can be initialized to 0.50. Once a certain state has been visited several times, the Q-coder arrives at a fairly accurate (within $q=1-p$, whichever is smaller) for that state.
 - » By updating the quantized probability estimate in any subsequent visits, the Q-coder can cope with changes in the source statistics.
 - » The probability estimation technique is based on the interval renormalization that is a necessary part of the finite-precision arithmetic coding process

IBM Q-Coder



- **Features:**

- **When the Q-coder is used to encode a stationary binary DMS (single context encoding), its output bit rate is about 5 to 6% above the source entropy, depending on the value of**
 - » **This is due partly restricted set of available probability estimates, partly to fluctuations in the probability estimation process, and partly to the approximations used in the interval partitioning process**
 - » **Multiple-context encoding results in an additional 1% inefficiency on the average**

IBM Q-Coder




- **Sequence of source symbols must be translated into a binary sequence when using the Q-coder.**
 - Since the probabilities of each binary symbol may depend on its neighbor, define a number of distinct conditioning states for which probability estimates need to be maintained.
 - Each conditioning state is referred to as a context
 - » With a Markov source define a separate context for each state of the Markov source
 - » The context does not correspond to an individual state, the states with the same probability are lumped together

Entropy Estimation




- **The most fundamental measure used in data compression is entropy**
- **How much lossless compression is theoretically possible depends on:**
 - the statistics of the data,
 - Overhead of the code,
- **Approach to estimating entropy:**
 - Characterize the source using a certain model, and then find the entropy with respect to that model.
 - Segment the data into blocks of size N and use the frequency of occurrence of each block as a measure of the probability

Entropy Estimation

- 
- **Source model accuracy is essential**
 - **Performance bounds are established by the entropy with respect to the model.**
 - **effectiveness of a model is determined by how accurately it predicts the symbol probabilities.**
 - **Real world data has a complex structure that is not modeled well.**
 - **The more complex the model, the closer the code will follow the actual entropy.**
 - **Need a code that implements a model that closely follows the data**

Entropy Estimation

- 
- **Block estimation will theoretically approach the entropy of the original source as N (the size of the block) grows to infinity.**
 - **Convergence is slow, and N must be large for the estimate to be accurate.**
 - **The number of combinations of sample of block N for 8 bit sample data is 256^N**
 - Computing the entropy using block estimation is practical for small values of N .

Text Compression Entropy

- A statistical model for the English language is based on the probabilities of the sample data set.
- If single letters are used:

Symbol	Probability	Symbol	Probability	Symbol	Probability
Space	0.1859	I	0.0575	R	0.0484
A	0.0642	J	0.0008	S	0.0514
B	0.0127	K	0.0049	T	0.0796
C	0.0218	L	0.0321	U	0.0228
D	0.0317	M	0.0198	V	0.0083
E	0.1031	N	0.0574	W	0.0175
F	0.0208	O	0.0632	X	0.0013
G	0.0152	P	0.0152	Y	0.0164
H	0.0467	Q	0.0008	Z	0.0005

Simple Text Model

- **If we assume equiprobable symbols:**

$$p(s_i) = 1 / 27, \text{ for } i = 1, \dots, 27$$

- **The entropy of this source is:**

$$H(S) = \log_2(27) = 4.75 \text{ bits / symbol}$$

- **A typical text generated by such a model is :**

- **ZEWRTZYNSADXESYJRQY*WGECIJJ*OBVKRBQPOZBY
MBUAWVLBTQCNIFMP*KMVUUGBSAXHLHSIE*M**
- **Model does not reflect any of the structure found in English**
- **The model is also called static context free**

Model Based On Observed Probabilities

- Model can be constructed using probabilities observed from the data
- A Discrete Memoryless Source (DMS) has the entropy:

$$H(S) = \sum_s p(s_i) \log_2 \left(\frac{1}{p(s_i)} \right) = 4.03 \text{ bits / symbol}$$

- Typical text generated by such a model would be:
 - AINGAI**ITF*NNR*ASAEV*OIE*BAINTHA*HYROO*POE
R*SETRYGAIETRWCO**EHDUARU*EU*C*FT*NSREM*D
IY*EESE**F*O*SRIS*R**UNNAS
 - Although the model produces more realistic proportions of vowel and consonants than the equiprobable model, it does not take into account the dependence among the different letters.
 - This is also called an “order-0 fixed context model”, the 0 refers to the number of preceding characters used in the model

First-Order Markov Source

- Simple model that will support dependence of successive symbols is a **First-Order Markov Source**
 - Need 27 probability tables similar to the single symbol/probability table.
 - One for each state of the Markov source determined by the preceding letter.
- The entropy of the first-order Markov model is:

$$H(S) = \sum_s \sum_s p(s_i, s_j) \log_2 \left(\frac{1}{p(s_i | s_j)} \right) = 3.32 \text{ bits / symbol}$$

- Typical text generated by such a model:
 - URTESHETHING*AD*E*AT*FOULE*ITHALIORT*WACT*D*STE*MINTS
AN*OLINS*TWID*OULY*TE*THIGHE*CO*YS*TH*HR*UPAVIDE*PAD*
CTAVED

Second-Order Markov Source

- **Second-Order Markov Source involves 729 (27x27) probability tables.**
 - One for each combination of the previous two letters.

- **Entropy of the second order model:**

$$H(S) = \sum_S \sum_S \sum_S p(s_i, s_j, s_k) \log_2 \left(\frac{1}{p(s_i | s_j, s_k)} \right) = 3.1 \text{ bits / symbol}$$

- **Typical text generated from the model:**
 - IANKS*CAN*OU*ANG*RLER*THATTED*OF*TO*SHOR*OF
*TO*HAVEMEM*A*I*MAND*AND*BUT*WHISSITABLY*TH
ERVEREER*EIGHTS*TAKILLIS*TA
- **Encoding text based on this model results in a bit rate close to 3.1 bits/symbol as compared to 4.75 bits/symbol for the equiprobable model.**

Shannon Estimate for English Entropy



- **Shannon exploited the fact that anyone speaking a language possesses an enormous knowledge of the language.**
- **Shannon found upper and lower bounds to the entropy of printed English by eliciting knowledge of the conditional probability distribution of the symbols from a subject through the use of a guessing game.**
 - **Subject was shown $N-1$ consecutive symbols of an unfamiliar text, and was asked to guess the next letter in the passage.**
 - **Guesses continued until the correct letter was selected.**
 - **The guessing process ranks the possible choices in decreasing order of conditional probability based on the subjects knowledge of the English language.**
 - **Experiment is run n times.**

Shannon Estimate Formula


- Denoting q_i^N the number of times the subject required i guesses to discover the correct letter given the previous $N-1$ letters.

- Shannon's entropy estimate:

$$\sum_{i=1}^{27} i \left(\frac{q_i^N}{n} - \frac{q_{i+1}^N}{n} \right) \log_2 i \leq H(S) \leq - \sum_{i=1}^{27} \frac{q_i^N}{n} \log_2 \left(\frac{q_i^N}{n} \right)$$

- 100 samples of English text were selected at random from a book, each 100 letters long ($N=100$).
- Shannon arrived at an upper bound of 1.3 bits/symbol and lower bound of 0.6 bits/symbol.

Sources of Error

- 
- **N is finite, does not reflect complete information regarding the past.**
 - **Probability of getting the right answer in the third guess was different in all those cases when the subject had to make three guesses**
 - **The sample size is finite, the experiment must be repeated many times before $\frac{q_i^N}{n}$ converges to its mean value**
 - **This estimate has not been repeatable by different authors, and subjects.**

Performance Measures - Lossless Compression



- **Lossless Compression has a theoretical limit (In terms of Bits/Symbol)**
- **Compression systems have overhead that keep the final size larger than the theoretical limit.**
- **Model limitations also keep the final size larger than the theoretical limits**
- **Ratio of Compression (Ratio of Uncompressed Data Set to Compressed Data Set)**

Performance Measures - Lossless Compression



- **Applications Mix Performance**
- **How close the data matches the underlying model of the compression code will be measured by the Ratio of Compression.**
- **A different selection of data with different statistics can build a execution profile**
- **The profile can be used to distinguish the performance of algorithms.**