IE404
Digital Image Processing

Dr. Manish Khare

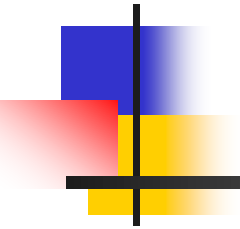


Image and Video Compression

Introduction and Overview

- ✓ The field of image compression continues to grow at a rapid pace

- ✓ As we look to the future, the need to store and transmit images will only continue to increase faster than the available capability to process all the data

- ✓ Applications that require image compression are many and varied such as:

1. Internet,
2. Businesses,
3. Multimedia,
4. Satellite imaging,
5. Medical imaging

- ✓ Compression algorithm development starts with applications to two-dimensional (2-D) still images
- ✓ After the 2-D methods are developed, they are often extended to video (motion imaging)
- ✓ However, we will focus on image compression of single frames of image data
- ✓ *Image compression* involves reducing the size of image data files, while *retaining necessary information*
- ✓ Retaining necessary information depends upon the application
- ✓ Image segmentation methods, which are primarily a data reduction process, can be used for compression

- Data compression: process of reducing the amount of data required to represent a given quantity of information
- Data and information are not the same.
- Data is used to convey information.
- Various amount of data may be used to represent the same information

- ✓ The reduced file created by the compression process is called the *compressed file* and is used to reconstruct the image, resulting in the *decompressed image*
- ✓ The original image, before any compression is performed, is called the *uncompressed image file*
- ✓ The ratio of the original, uncompressed image file and the compressed file is referred to as the *compression ratio*

- ✓ The compression ratio is denoted by:

$$Compression\ Ratio = \frac{Uncompressed\ \ file\ size}{Compressed\ \ file\ size} = \frac{SIZE_U}{SIZE_C}\ ;\ Often\ written\ as \rightarrow SIZE_U : SIZE_C$$

EXAMPLE 10.1.1: The original image is 256x256 pixels, single-band (grayscale), 8-bits per pixel. This file is 65,536 bytes (64k). After compression the image file is 6,554 bytes. The compression ratio is: $SIZE_U/SIZE_C = 65536/6554 = 9.999 \approx 10$. This can also be written as 10:1

This is called a "10 to 1 compression", a "10 times compression", or can be stated as

"compressing the image to 1/10 its original size". Another way to state the compression is to use

the terminology of *bits per pixel*. For an NxN image:

$$Bits\ per\ pixel = \frac{Number\ of\ bits}{Number\ of\ pixels} = \frac{(8)(Number\ of\ bytes)}{NxN}$$

EXAMPLE 10.1.2: Using the preceding example, with a compression ratio of 65,536/6,554 bytes, we want to express this as bits per pixel. This is done by first finding the number of pixels in the image: 256x256 = 65,536 pixels. We then find the number of bits in the compressed image file: (6,554 bytes)(8 bits/byte) = 52,432 bits. Now we can find the bits per pixel by taking the ratio: 52,432/65,536 = 0.8 bits/pixel

- ✓ The reduction in file size is necessary to meet the bandwidth requirements for many transmission systems, and for the storage requirements in computer databases

- ✓ Also, the amount of data required for digital images is enormous

EXAMPLE 10.1.3:

To transmit an RGB (color) 512x512, 24-bit (8-bit per pixel per color) image via modem at 56 kbaud (kilo-bits per second), it would take about:

$$\frac{(512 \times 512 \text{ pixels})(24 \text{ bits/pixel})}{(56 \times 1024 \text{ bits/second})} \approx 109 \text{ seconds} \approx 1.8 \text{ minutes}$$

- ✓ This number is based on the actual transmission rate being the maximum, which is typically not the case due to Internet traffic, overhead bits and transmission errors

- ✓ Additionally, considering that a web page might contain more than one of these images, the time it takes is simply too long

- ✓ For high quality images the required resolution can be much higher than the previous example

EXAMPLE 10.1.4:

To transmit a digitized color 35mm slide scanned at 4000x3000 pixels, and 24-bits, at 56 kbaud would take about:

$$\frac{(4000 \times 3000 pixels)(24 bits/pixel)}{(56 \times 1024 bits/sec)} \approx 5022 sec \approx 83 min$$, too long to wait!

Example 10.1.5 applies maximum data rate to Example 10.1.4

EXAMPLE 10.1.5:

To transmit a digitized color 35mm slide scanned at 4000x3000 pixels, and 24-bits, at 3 Mbps would take about:

$$\frac{(4000 \times 3000 \text{ pixels})(24 \text{ bits/pixel})}{(3 \times 1024 \times 1024 \text{ bits/second})} \approx 91 \text{ seconds} \approx 1.5 \text{ minutes}$$

- ✓ Now, consider the transmission of video images, where we need multiple frames per second
- ✓ If we consider just one second of video data that has been digitized at 640x480 pixels per frame, and requiring 15 frames per second for interlaced video, then:

EXAMPLE 10.1.6: To transmit one second of interlaced video that has been digitized at 640x480 pixels:

$$\frac{(640 \times 480 \times 15\, frames/\text{sec})(24 bits/pixel)}{3 \times 1024 \times 1024 bits/\text{sec})} \approx 35\, \text{sec}onds$$

- ✓ Waiting 35 seconds for one second's worth of video is not exactly real time!
- ✓ Even attempting to transmit uncompressed video over the highest speed Internet connection is impractical
- ✓ For example: The Japanese Advanced Earth Observing Satellite (ADEOS) transmits image data at the rate of 120 Mbps

- ✓ Applications requiring high speed connections such as high definition television, real-time teleconferencing, and transmission of multiband high resolution satellite images, leads us to the conclusion that *image compression is not only desirable but necessary*

- ✓ *Key to a successful compression scheme* is *retaining necessary information*

- ✓ To understand "*retaining necessary information*", we must differentiate between *data* and *information*

1. *Data*:

- For digital images*, data* refers to the pixel gray level values that correspond to the brightness of a pixel at a point in space
- Data are used to convey information, much like the way the alphabet is used to convey information via words

2. *Information:*

- Information is an interpretation of the data in a meaningful way

- Information is an elusive concept; it can be application specific

- ✓ There are two primary types of image compression methods:

1. *Lossless compression methods:*

- Allows for the exact recreation of the original image data, and can compress complex images to a maximum 1/2 to 1/3 the original size – 2:1 to 3:1 compression ratios
- Preserves the data *exactly*

2. *Lossy compression methods:*

- Data loss, original image cannot be re-created exactly

- Can compress complex images 10:1 to 50:1 and retain high quality, and 100 to 200 times for lower quality, but acceptable, images

Fundamentals

- Suppose that two representations have the same amount of information one using n1 units of data and the other one n2 units of data.
- Compression ratio: C_R=n1/n2
- Relative data redundancy of the first set is defined as:
- R_D=1-1/ C_R
- n2<<n1, C_R approaches infinity and R_D approaches one. This means that the data n1 has been highly redundant.
- n2>>n1, C_R approaches zero and R_D approaches minus infinity. This means that the data n2 contains more data (expansion instead of compression).

Fundamentals

- A typical compression ratio: 10
- C_R=10, R_D=1-1/ 10=0.9
- 90% of the data in the first set is redundant

- ✓ Compression algorithms are developed by taking advantage of the redundancy that is inherent in image data

- ✓ Four primary types of redundancy that can be found in images are:

1. *Coding*
2. *Interpixel*
3. *Interband*
4. *Psychovisual redundancy*

Data Redundancy

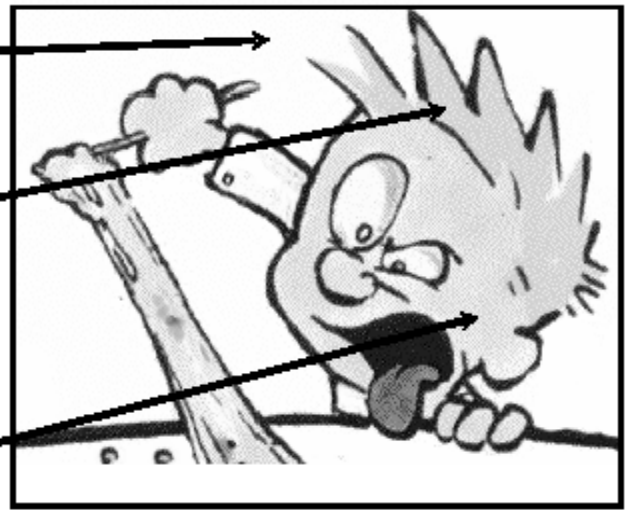
Three basic data redundancies

- Coding Redundancy
- Interpixel Redundancy
- Psychovisual Redundancy

CR: some graylevels are more common than others

IR: the same graylevel covers large areas

PVR: the eye can only resolve 32 graylevels locally



1. *Coding redundancy*

- ✓ Occurs when the data used to represent the image is not utilized in an optimal manner

2. *Interpixel redundancy*

- ✓ Occurs because adjacent pixels tend to be highly correlated, in most images the brightness levels do not change rapidly, but change gradually

3. *Interband redundancy*

✓ Occurs in color images due to the correlation between bands within an image – if we extract the red, green and blue bands they look similar

4. *Psychovisual redundancy*

✓ Some information is more important to the human visual system than other types of information

Coding Redundancy

A natural m-bit coding method assigns m-bit to each gray level without considering the probability that gray level occurs with: Very likely to contain coding redundancy

Basic concept:

- Utilize the probability of occurrence of each gray level (histogram) to determine length of code representing that particular gray level: variable-length coding.
- Assign shorter code words to the gray levels that occur most frequently or vice versa.

Coding Redundancy

Let $0 \le r_k \le 1$: gray levels (discrete random variable)

$p_r(r_k)$: probability of occurrence of r_k

n_k : number of pixels that r_k appears in the image

n : total number of pixels in an image

L : number of gray levels

$l(r_k)$: number of bits used to represent r_k

L_{avg} : average length of code words assigned to the grey levels

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) \quad \text{where} \quad p_r(r_k) = \frac{n_k}{n}, k = 0,1,\ldots\ldots, L-1$$

Hence, total number of bits required to code an $M \times N$ image is MNL_{avg}

For a natural m-bit coding $L_{avg} = m$.

Coding Redundancy

Let us assume an 8-level image

8-levels: can be represented by 3-bits, m = 3.

Normalized histogram of the image is given by:

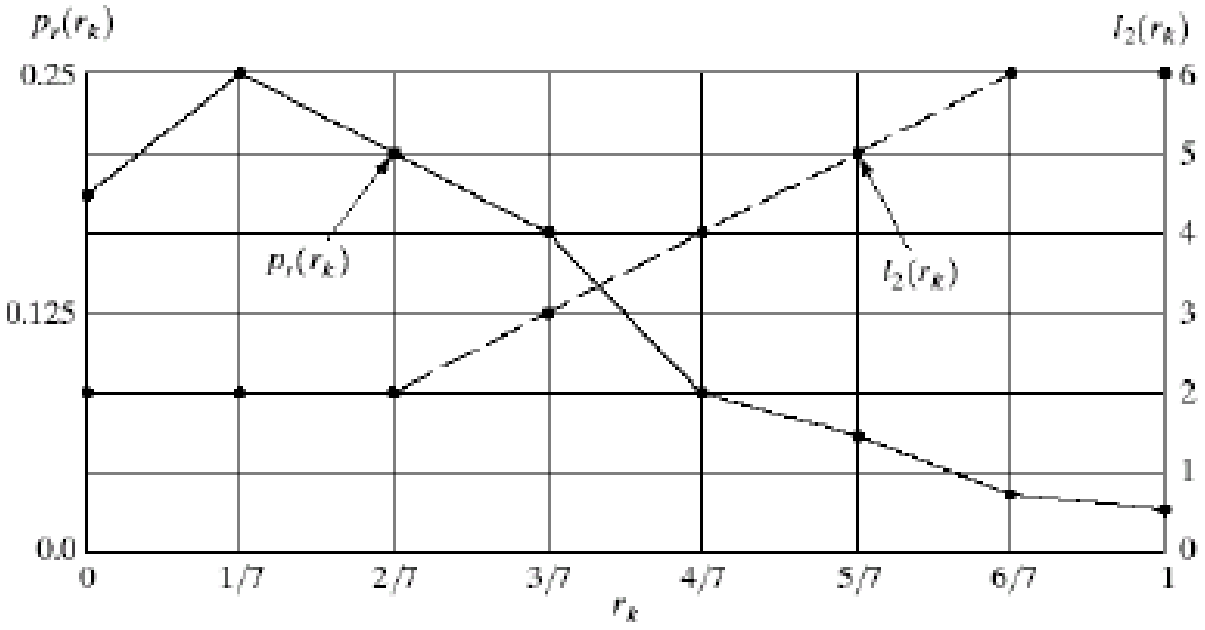


FIGURE 8.1 Graphic representation of the fundamental basis of data compression through variable-length coding.

Coding Redundancy

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

TABLE 8.1 Example of variable-length coding.

$$L_{avg_1} = m = 3 \text{ bits}, \quad \text{and}$$

$$L_{avg_2} = \sum_{k=0}^{7} l_2(r_k) p_r(r_k)$$

$$= 2(0.19) + 2(0.25) + 2(0.31) + 3(0.16)$$

$$+ 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02) = 2.7 \text{ bits}$$

Coding Redundancy

Example 3-bit image:

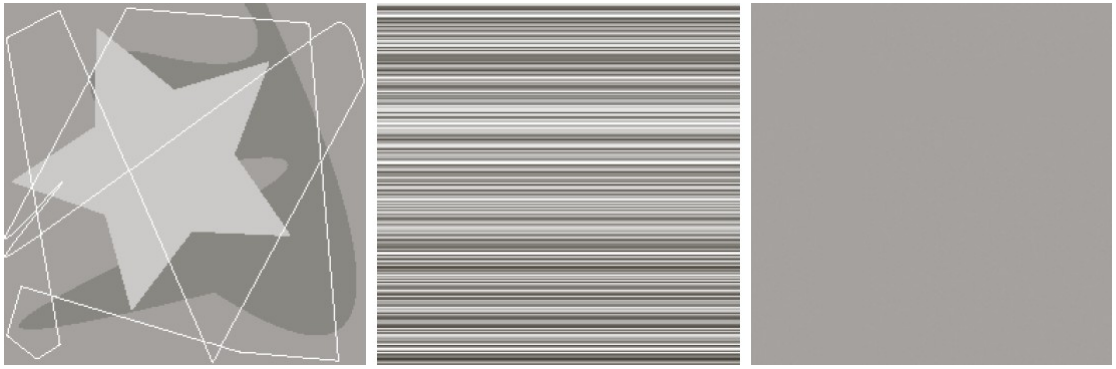
gray level r_k	probability $p(r_k)$	source	code
0	0.1	000	01
1	0.4	001	0
2	0.03	010	11
3	0.05	011	10
4	0.3	100	1
5	0.1	101	00
6	0.01	110	111
7	0.01	111	000

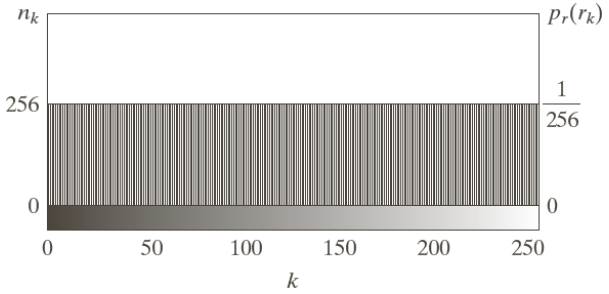
$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)\,p(r_k)$$

source : L_{avg}=(constant $l(r_k)$=3)=3*1=3

code: L_{avg}=0.1*2+0.4*1+…=1.32

Spatial and temporal redundancy





Spatial and temporal redundancy

- Codes designed based on histogram do not exploit the correlation between pixels.
- These correlations result from the structural or geometrical relationships between the objects in the image.
- The value of any pixel is highly correlated to its neighbors and the information carried by individual pixels is small.

Spatial and temporal redundancy

- How to reduce spatial and temporal redundancy?
- The image (which is a 2-D array) is transformed into a more efficient format.
- This process is called mapping or transformations.
- The transformation should reduce the redundancy.
- Example: the transform could be the difference between adjacent pixels
- The transform (or mapping) should be reversible. The original image should be recoverable from the transform.

Irrelevant Information

- The eye does not response with equal sensitivity to all visual information
- Certain information has less relative importance than other
- This information is said to be psychovisually redundant. It can be eliminated without significantly impairing the quality of image perception.
- Since the elimination of irrelevant information results in a loss of quantitative information, it is commonly referred to as quantization.

- ✓ The key in image compression algorithm development is to determine the minimal data required to retain the necessary information
- ✓ The compression is achieved by taking advantage of the redundancy that exists in images
- ✓ If the redundancies are removed prior to compression, for example with a decorrelation process, a more effective compression can be achieved

- ✓ To help determine which information can be removed and which information is important, the *image fidelity criteria* are used
- ✓ These measures provide metrics for determining image quality
- ✓ It should be noted that the information required is application specific, and that, with lossless schemes, there is no need for a fidelity criteria

Interpixel Redundancy

- Caused by High Interpixel Correlations within an image, i.e., gray level of any given pixel can be reasonably predicted from the value of its neighbors (information carried by individual pixels is relatively small) spatial redundancy, geometric redundancy, interframe redundancy (in general, interpixel redundancy)
- To reduce the interpixel redundancy, mapping is used. The mapping scheme can be selected according to the properties of redundancy.
- An example of mapping can be to map pixels of an image: $f(x,y)$ to a sequence of pairs: (g_1,r_1), (g_2,r_2), ..., (g_i,r_i), ..

g_i: ith gray level r_i: run length of the ith run

Measuring Image Information

A random event E that occurs with probability P(E) is said to contain :

$$I(E) = \log \frac{1}{P(E)} = -\log P(E)$$

units of information

I(E) self-information

If P(E)=1 then I(E)=0,
no uncertainty is associated with the event, no information would be transformed by communicating that E has occurred.

Fidelity Criteria

- Since information may be lost during compression a means of quantifying the information loss is desirable.
- 2 type of criteria:
  1. Objective fidelity criteria
  2. Subjective fidelity criteria
- Objective fidelity criteria: the information loss is expressed as a function of input image (original image) and output image (compressed and decompressed)

Fidelity Criteria

- Root mean square (rms) error:

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2 \right]^{1/2}$$

Mean square signal to noise ratio:

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\hat{f}(x,y) \right)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2}$$

Fidelity Criteria

- Objective fidelity criteria: simple and convenient, sometimes misleading
- Subjective fidelity criteria: measuring image quality by the subjective evaluation of a group of viewers and averaging their evaluations.

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

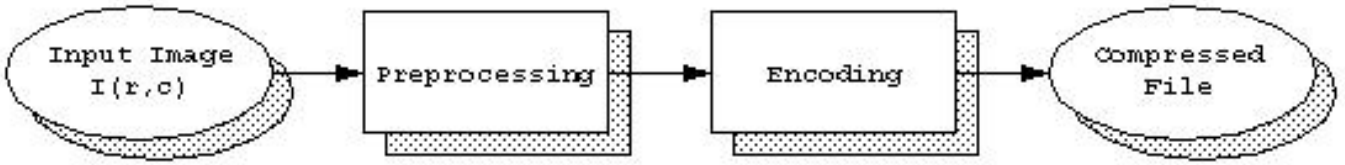
✓ Compression System Model

- The compression system model consists of two parts:

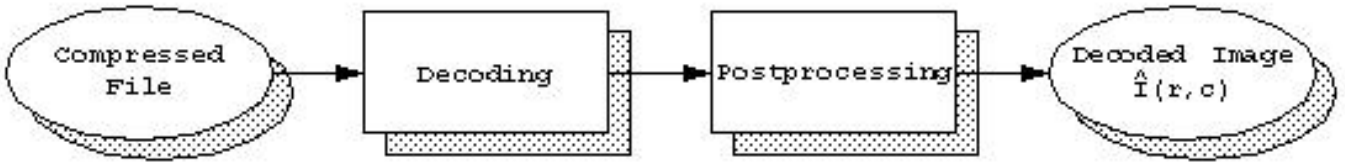
1. The compressor
2. The decompressor

- The *compressor* consists of a preprocessing stage and encoding stage, whereas the *decompressor* consists of a decoding stage followed by a postprocessing stage

Figure 10.1-1: Compression System Model



a) Compression



b) Decompression

- Before encoding, preprocessing is performed to prepare the image for the encoding process, and consists of any number of operations that are application specific

- After the compressed file has been decoded, postprocessing can be performed to eliminate some of the potentially undesirable artifacts brought about by the compression process

- The compressor can be broken into following stages:

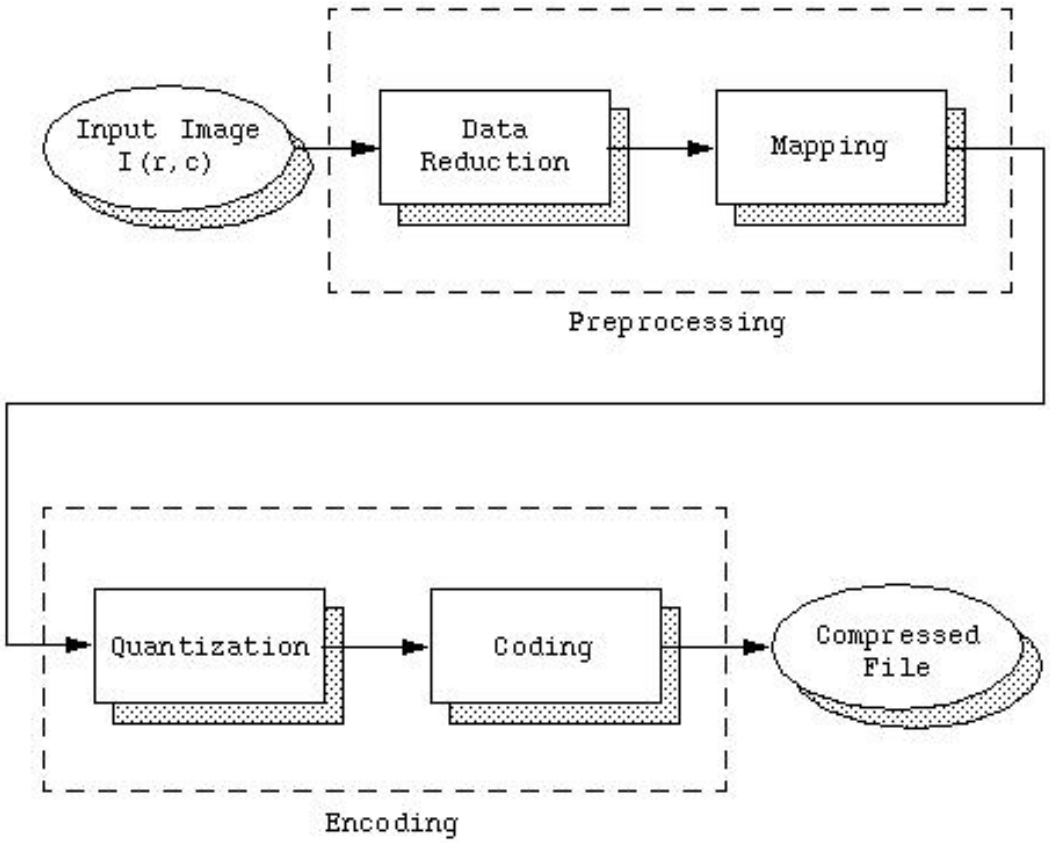
1. *Data reduction*: Image data can be reduced by gray level and/or spatial quantization, or can undergo any desired image improvement (for example, noise removal) process
2. *Mapping*: Involves mapping the original image data into another mathematical space where it is easier to compress the data

3. *Quantization*: Involves taking potentially continuous data from the mapping stage and putting it in discrete form

4. *Coding*: Involves mapping the discrete data from the quantizer onto a code in an optimal manner

- A compression algorithm may consist of all the stages, or it may consist of only one or two of the stages

Figure 10.1-2: The Compressor



- The decompressor can be broken down into following stages:

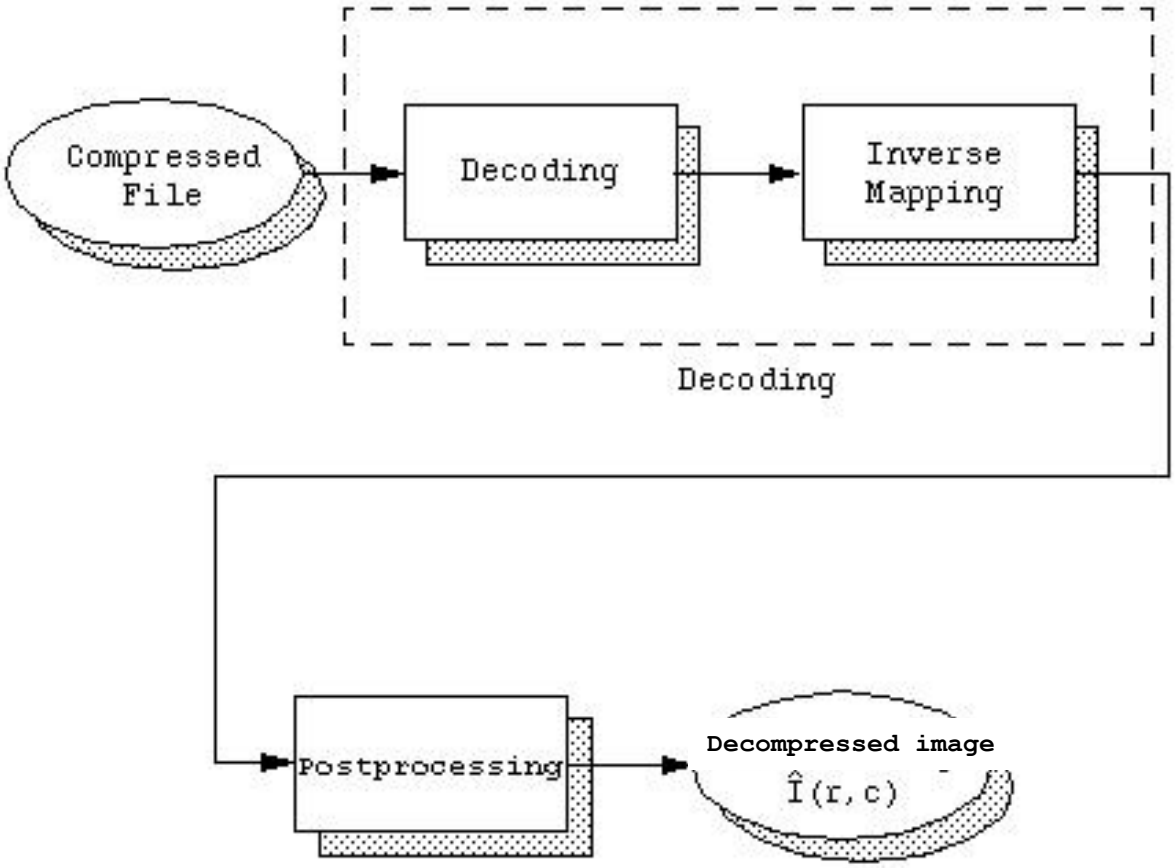
1. *Decoding*: Takes the compressed file and reverses the original coding by mapping the codes to the original, quantized values

2. *Inverse mapping*: Involves reversing the original mapping process

3. *Postprocessing*: Involves enhancing the look of the final image

- This may be done to reverse any preprocessing, for example, enlarging an image that was shrunk in the data reduction process
- In other cases the postprocessing may be used to simply enhance the image to ameliorate any artifacts from the compression process itself

Figure 10.1-3: The Decompressor



- The development of a compression algorithm is highly application specific
- Preprocessing stage of compression consists of processes such as enhancement, noise removal, or quantization are applied
- The goal of preprocessing is to prepare the image for the encoding process by eliminating any irrelevant information, where *irrelevant* is defined by the application

- For example, many images that are for viewing purposes only can be preprocessed by eliminating the lower bit planes, without losing any useful information

Figure - Bit plane images

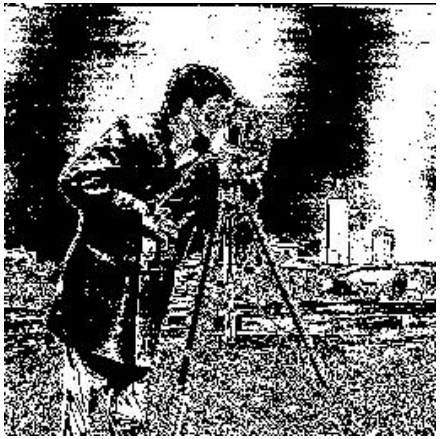
a) Original image

b) Bit plane 7, the most significant bit

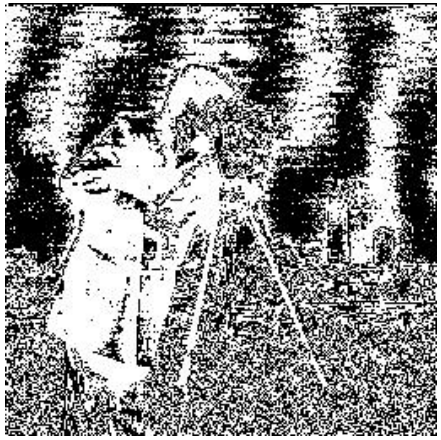
c) Bit plane 6

d) Bit plane 5

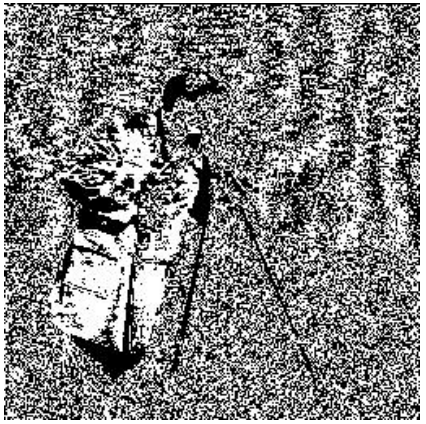
Figure 10.1.4 Bit plane images (Contd)



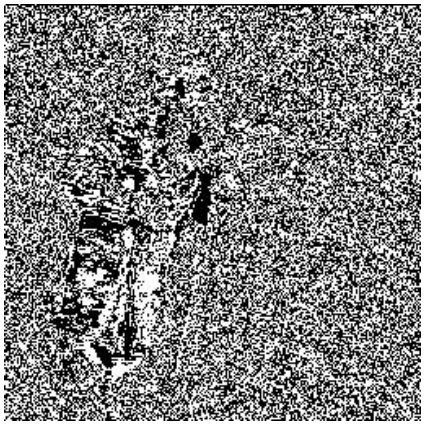
e) Bit plane 4



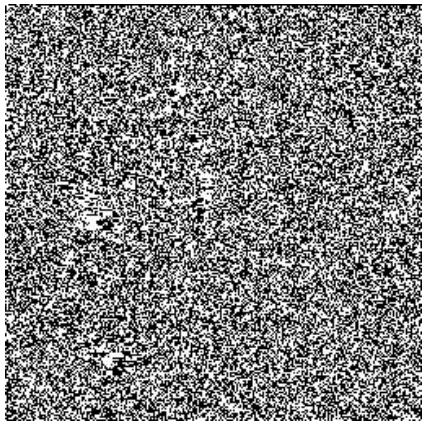
f) Bit plane 3



g) Bit plane 2



h) Bit plane 1



i) Bit plane 0,
the least significant bit

- The mapping process is important because image data tends to be highly correlated
- Specifically, if the value of one pixel is known, it is highly likely that the adjacent pixel value is similar
- By finding a mapping equation that decorrelates the data this type of data redundancy can be removed

- *Differential coding:* Method of reducing data redundancy, by finding the difference between adjacent pixels and encoding those values

- The *principal components transform* can also be used, which provides a theoretically optimal decorrelation

- *Color transforms* are used to decorrelate data between image bands

- *Quantization* may be necessary to convert the data into digital form (BYTE data type), depending on the mapping equation used

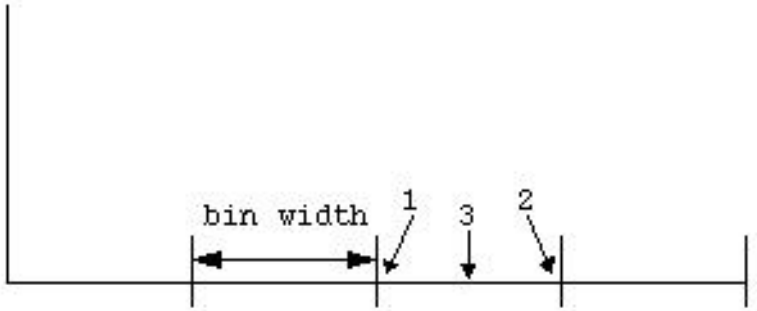
- This is because many of these mapping methods will result in floating point data which requires multiple bytes for representation which is not very efficient, if the goal is data reduction

- Quantization can be performed in the following ways:

1. *Uniform quantization:* In it, all the quanta, or subdivisions into which the range is divided, are of equal width

2. *Nonuniform quantization:* In it the quantization bins are not all of equal width

Figure 3.2-18: QUANTIZATION BINS



a) Uniform quantization bins: all bins are the same width. Values that fall within the same bin can be mapped to the low end(1), high end(2), or the middle(3).

b) Variable quantization bins are of different widths.

- Often, nonuniform quantization bins are designed to take advantage of the response of the human visual system

- In the spectral domain, the higher frequencies may also be quantized with wider bins because we are more sensitive to lower and midrange spatial frequencies and most images have little energy at high frequencies

- The concept of nonuniform quantization bin sizes is also described as a *variable bit rate*, since the wider quantization bins imply fewer bits to encode, while the smaller bins need more bits

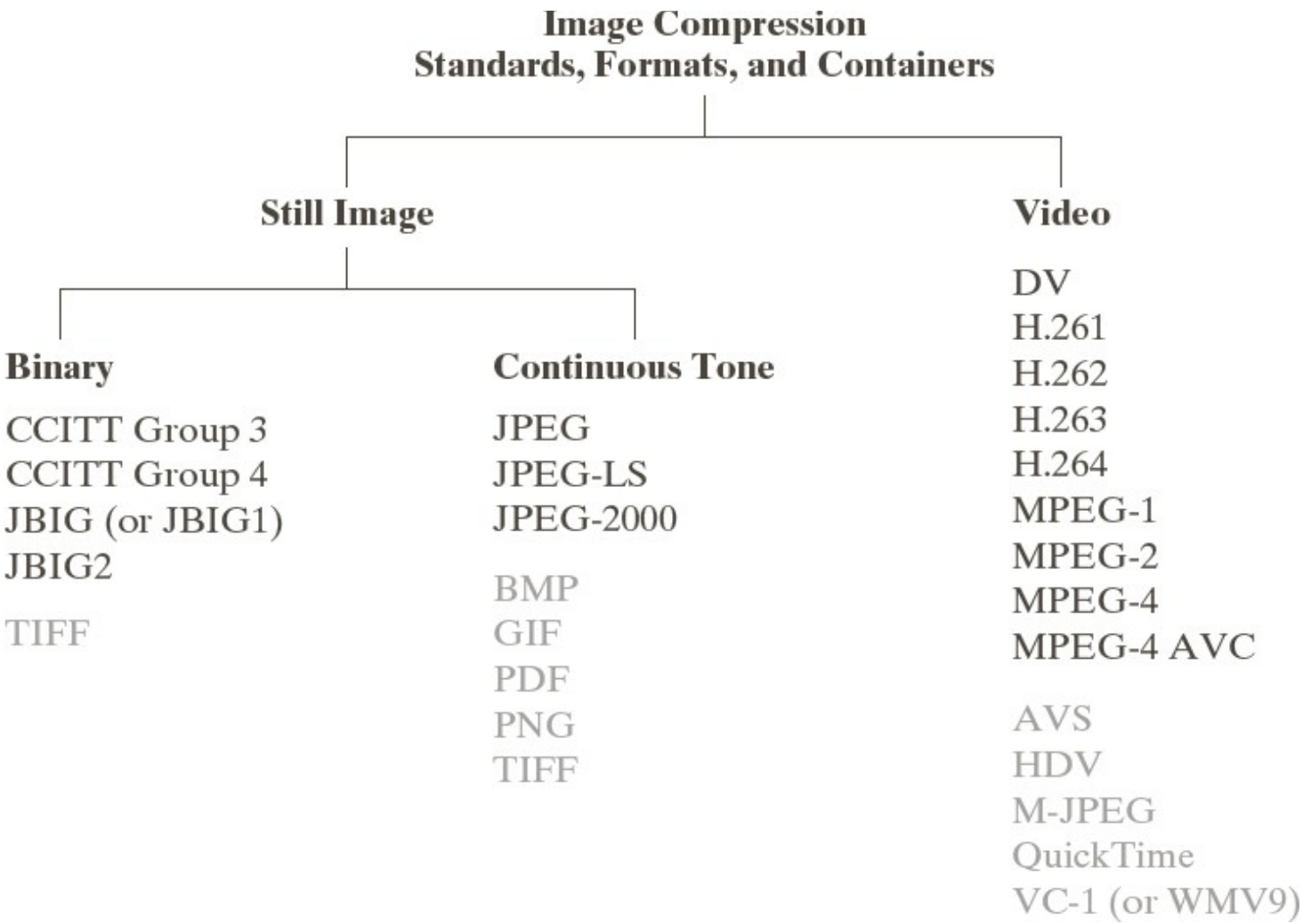
- It is important to note that the quantization process is not reversible, so it is not in the decompression model and also some information may be lost during quantization

- The coder in the coding stage provides a one-to-one mapping, each input is mapped to a unique output by the coder, so it is a reversible process

- The code can be an *equal length code,* where all the code words are the same size, or an *unequal length code* with variable length code words

- In most cases, an unequal length code is the most efficient for data compression, but requires more overhead in the coding and decoding stages

Image Formats, Compression Standards



Name	Organization	Description
Bi-Level Still Images		
CCITT Group 3	ITU-T	Designed as a facsimile (FAX) method for transmitting binary documents over telephone lines. Supports 1-D and 2-D run-length [8.2.5] and Huffman [8.2.1] coding.
CCITT Group 4	ITU-T	A simplified and streamlined version of the CCITT Group 3 standard supporting 2-D run-length coding only.
JBIG *or* JBIG1	ISO/IEC/ ITU-T	A *Joint Bi-level Image Experts Group* standard for progressive, lossless compression of bi-level images. Continuous-tone images of up to 6 bits/pixel can be coded on a bit-plane basis [8.2.7]. Context sensitive arithmetic coding [8.2.3] is used and an initial low resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary based methods [8.2.6] for text and halftone regions, and Huffman [8.2.1] or arithmetic coding [8.2.3] for other image content. It can be lossy or lossless.
Continuous-Tone Still Images		
JPEG	ISO/IEC/ ITU-T	A *Joint Photographic Experts Group* standard for images of photographic quality. Its lossy *baseline coding system* (most commonly implemented) uses quantized discrete cosine transforms (DCT) on 8×8 image blocks [8.2.8], Huffman [8.2.1], and run-length [8.2.5] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ ITU-T	A lossless to near-lossless standard for continuous tone images based on adaptive prediction [8.2.9], context modeling [8.2.3], and Golomb coding [8.2.2].
JPEG-2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.2.3] and quantized discrete wavelet transforms (DWT) [8.2.10] are used. The compression can be lossy or lossless.

(*Continues*)

Name	Organization	Description
Video		
DV	IEC	*Digital Video*. A video standard tailored to home and semiprofessional video production applications and equipment—like electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.2.8] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (*integrated services digital network*) lines. It supports non-interlaced 352 × 288 and 176 × 144 resolution images, called CIF (*Common Intermediate Format*) and QCIF (*Quarter CIF*), respectively. A DCT-based compression approach [8.2.8] similar to JPEG is used, with frame-to-frame prediction differencing [8.2.9] to reduce temporal redundancy. A block-based technique is used to compensate for motion between frames.
H.262	ITU-T	See MPEG-2 below.
H.263	ITU-T	An enhanced version of H.261 designed for ordinary telephone modems (i.e., 28.8 Kb/s) with additional resolutions: SQCIF (*Sub-Quarter* CIF 128 × 96), 4CIF (704 × 576), and 16CIF (1408 × 512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, Internet streaming, and television broadcasting. It supports prediction differences within frames [8.2.9], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.2.3].
MPEG-1	ISO/IEC	A *Motion Pictures Expert Group* standard for CD-ROM applications with non-interlaced video at up to 1.5 Mb/s. It is similar to H.261 but frame predictions can be based on the previous frame, next frame, or an interpolation of both. It is supported by almost all computers and DVD players.
MPEG-2	ISO/IEC	An extension of MPEG-1 designed for DVDs with transfer rates to 15 Mb/s. Supports interlaced video and HDTV. It is the most successful video standard to date.
MPEG-4	ISO/IEC	An extension of MPEG-2 that supports variable block sizes and prediction differencing [8.2.9] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 *Advanced Video Coding* (AVC). Identical to H.264 above.

Name	Organization	Description
Continuous-Tone Still Images		
BMP	Microsoft	*Windows Bitmap*. A file format used mainly for simple uncompressed images.
GIF	CompuServe	*Graphic Interchange Format*. A file format that uses lossless LZW coding [8.2.4] for 1- through 8-bit images. It is frequently used to make small animations and short low resolution films for the World Wide Web.
PDF	Adobe Systems	*Portable Document Format*. A format for representing 2-D documents in a device and resolution independent way. It can function as a container for JPEG, JPEG 2000, CCITT, and other compressed images. Some PDF versions have become ISO standards.
PNG	*World Wide Web Consortium* (W3C)	*Portable Network Graphics*. A file format that losslessly compresses full color images with transparency (up to 48 bits/pixel) by coding the difference between each pixel's value and a predicted value based on past pixels [8.2.9].
TIFF	Aldus	*Tagged Image File Format*. A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
Video		
AVS	MII	*Audio-Video Standard*. Similar to H.264 but uses exponential Golomb coding [8.2.2]. Developed in China.
HDV	Company consortium	*High Definition Video*. An extension of DV for HD television that uses MPEG-2 like compression, including temporal redundancy removal by prediction differencing [8.2.9].
M-JPEG	Various companies	*Motion JPEG*. A compression format in which each frame is compressed independently using JPEG.
Quick-Time	Apple Computer	A media container supporting DV, H.261, H.262, H.264, MPEG-1, MPEG-2, MPEG-4, and other video compression formats.
VC-1 WMV9	SMPTE Microsoft	The most used video format on the Internet. Adopted for HD and *Blu-ray* high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.2.8 and 8.2.9] and context dependent variable-length code tables [8.2.1]—but no predictions within frames.

➢LOSSLESS COMPRESSION METHODS

- ✓No loss of data, decompressed image exactly same as uncompressed image
- ✓Medical images or any images used in courts
- ✓Lossless compression methods typically provide about a 10% reduction in file size for complex images

- ✓ Lossless compression methods can provide substantial compression for simple images

- ✓ However, lossless compression techniques may be used for both preprocessing and postprocessing in image compression algorithms to obtain the extra 10% compression

- ✓ The underlying theory for lossless compression (also called *data compaction*) comes from the area of communications and information theory, with a mathematical basis in probability theory

- ✓ One of the most important concepts used is the idea of *information content* and *randomness in data*

- ✓ *Information theory* defines information based on the probability of an event, knowledge of an unlikely event has more information than knowledge of a likely event
- ✓ For example:
- The earth will continue to revolve around the sun; little information, 100% probability
- An earthquake will occur tomorrow; more info. Less than 100% probability
- A matter transporter will be invented in the next 10 years; highly unlikely – low probability, high information content

- ✓ This perspective on information is the *information theoretic definition* and should not be confused with our working definition that requires information in images to be useful, not simply novel

- ✓ *Entropy* is the measurement of the average information in an image

- ✓ The entropy for an *N x N* image can be calculated by this equation:

$$Entropy = -\sum_{i=0}^{L-1} p_i \log_2(p_i) \quad (in\ bits/pixel)$$

$$Where\ p_i = the\ probability\ of\ the\ i^{th}\ gray\ level = \frac{n_k}{N^2}$$

$$n_k = the\ total\ number\ of\ pixels\ with\ gray\ value\ k$$

$$L = the\ total\ number\ of\ gray\ levels\ (e.g.\ 256\ for\ 8\text{-}bits)$$

- ✓ This measure provides us with a theoretical minimum for the average number of bits per pixel that could be used to code the image

- ✓ It can also be used as a metric for judging the success of a coding scheme, as it is theoretically optimal

EXAMPLE 10.2.1:

Let L = 8, meaning there are 3 bits/pixel in the original image. Now, let's say the number of pixels at each gray level value is equal (they have the same probability), that is:

$$p_0 = p_1 = \ldots = p_7 = \frac{1}{8}$$

Now, we can calculate the entropy as follows:

$$Entropy = -\sum_{i=0}^{7} p_i \log_2(p_i) = -\sum_{i=0}^{7} \frac{1}{8} \log_2\left(\frac{1}{8}\right) = 3$$

This tells us that the theoretical minimum for lossless coding for this image is 3 bits per pixel. In other words, there is no code that will provide better results than the one currently used (called the natural code, since $000_2 = 0, 001_2 = 1, 010_2 = 2, \ldots, 111_2 = 7$). This example illustrates that the image with the most random distribution of gray levels, a uniform distribution, has the highest entropy

EXAMPLE 10.2.2:

Let = 8, thus we have a natural code with 3 bits per pixel in the original image. Now lets say that the entire image has a gray level of 2, so:

$$p_2 = 1, and\ p_0 = p_1 = p_3 = p_4 = p_5 = p_6 = p_7 = 0$$

And the entropy is:

$$Entropy = -\sum_{i=0}^{7} p_i \log_2(p_i) = -(1)\log_2(1) + 0 + \ldots + 0 = 0$$

This tells us the theoretical minimum for coding this image is 0 bits per pixel. Why is this? – Because the gray level value is known to be 2. To code the entire image we need only one value, this is called the certain event, it has a probability of 1

- ✓ The two preceding examples (10.2.1 and 10.2.2) illustrate the range of the entropy:

$$0 \leq Entropy \leq \log_2(L)$$

- ✓ The examples also illustrate the information theory perspective regarding information and randomness
- ✓ The more randomness that exists in an image, the more evenly distributed the gray levels, and more bits per pixel are required to represent the data

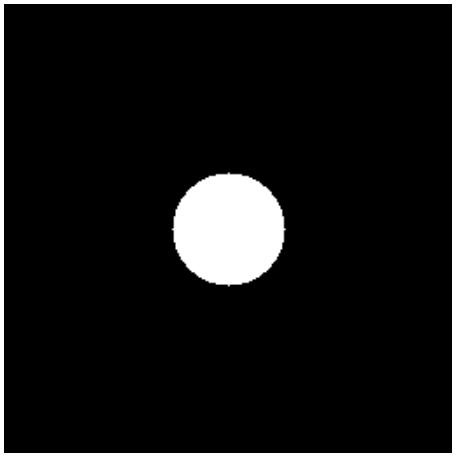
Figure 10.2-1 Entropy

a) Original image,
entropy = 7.032 *bpp*

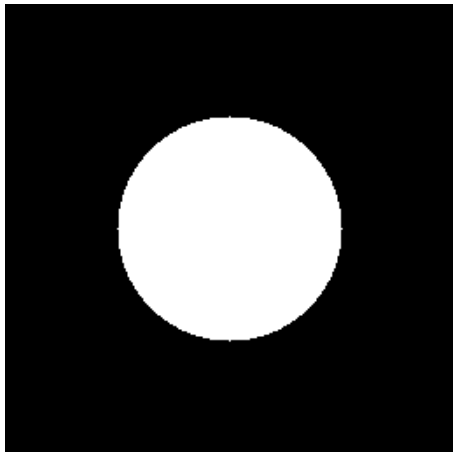
b) Image after local histogram equalization,
block size 4, entropy = 4.348 *bpp*

c) Image after binary threshold,
entropy = 0.976 *bpp*

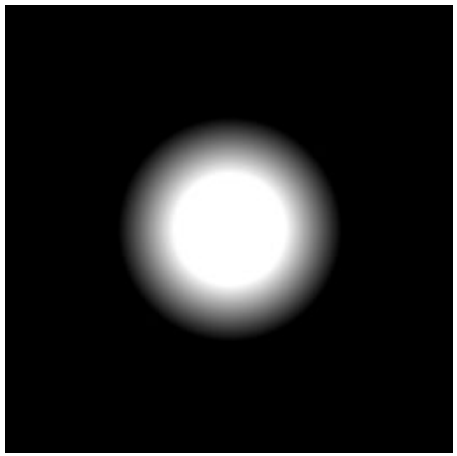
Figure 10.2-1 Entropy (contd)



d) Circle with a radius of 32,
entropy = 0.283 *bpp*



e) Circle with a radius of 64,
entropy = 0.716 *bpp*



f) Circle with a radius of 32,
and a linear blur radius of 64,
entropy = 2.030 *bpp*

- ✓ Figure 10.2.1 depicts that a minimum overall file size will be achieved if a smaller number of bits is used to code the most frequent gray levels
- ✓ Average number of bits per pixel (*L*ength) in a coder can be measured by the following equation:

$$L_{ave} = \sum_{i=0}^{L-1} l_i \; p_i$$

$$Where \; l_i = length \; in \; bits \; of \; the \; code \; for \; i^{th} \; gray \; level$$

$$p_i = histogram\text{-}probability \; of \; i^{th} \; gray \; level$$

✓ Huffman Coding

- The Huffman code, developed by D. Huffman in 1952, is a *minimum length code*
- This means that given the statistical distribution of the gray levels (the histogram), the Huffman algorithm will generate a code that is as close as possible to the *minimum bound, the entropy*

- The method results in an *unequal* (or *variable*) *length code*, where the size of the code words can vary

- For complex images, Huffman coding alone will typically reduce the file by 10% to 50% (1.1:1 to 1.5:1), but this ratio can be improved to 2:1 or 3:1 by preprocessing for irrelevant information removal

- The Huffman algorithm can be described in five steps:

1. Find the gray level probabilities for the image by finding the histogram
2. Order the input probabilities (histogram magnitudes) from smallest to largest
3. Combine the smallest two by addition
4. GOTO step 2, until only two probabilities are left
5. By working backward along the tree, generate code by alternating assignment of 0 and 1

EXAMPLE 10.2.3:

We have an image with 2-bits per pixel, giving four possible gray levels. The image is 10 rows by 10 columns.

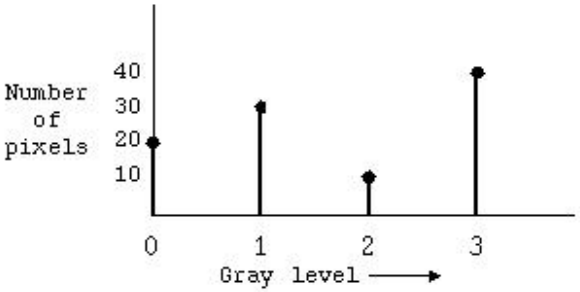
In Step 1 we find the histogram for the image. This is shown in Figure 10.2-2a, where we see that gray level 0 has 20 pixels, gray level 1 has 30 pixels, gray level 2 has 10 pixels, and gray level 3 has 40 pixels with the value. These are converted into probabilities by normalizing to the total number of pixels in the image.

Next, in step 2, the probabilities are ordered as in Figure 10.2-2b.

For step 3, we combine the smallest two by addition.

Step 4 repeats steps 2 and 3, where we reorder (if necessary) and add the two smallest probabilities as in Figure 10.2-2d. This step is repeated until only two values remain.

Figure 10.2-2: Huffman Coding Example



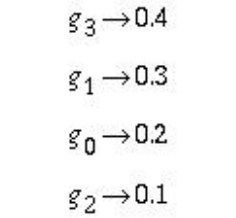
$g_0 = \frac{20}{100} = 0.2$

$g_1 = \frac{30}{100} = 0.3$

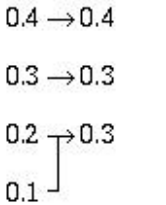
$g_2 = \frac{10}{100} = 0.1$

$g_3 = \frac{40}{100} = 0.4$

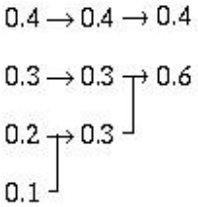
a) Step 1: Histogram

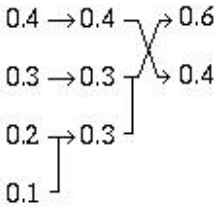


b) Step 2: Order



c) Step 3: Add





d) Step 4: Reorder and add until only two values remain

EXAMPLE 10.2.3 (Contd):

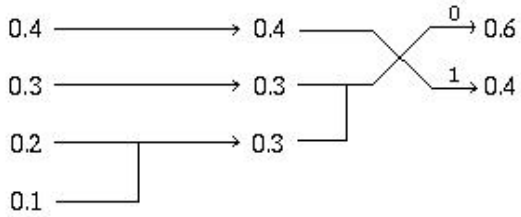
Since we have only two left in our example, we can continue to step 5 where the actual code assignment is made. The code assignment is shown in Figure 10.2-3. We start on the right-hand side of this tree and assign 0's and 1's, working our way back to the original probabilities.

Figure 10.2-3a shows the first assignment of 0 and 1. A 0 is assigned to the 0.6 branch, and a 1 to the 0.4 branch. In Figure 10.2-3b, the assigned 0 and 1 are brought back along the tree, and wherever a branch occurs the code is put on both branches.

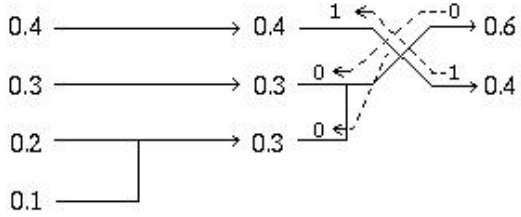
Now (Figure 10.2-3c), we assign the 0 and 1 to the branches labeled 0.3, appending to the existing code. Finally (Figure 10.2-3d), the codes are brought back one more level, and where the branch splits another assignment of 0 and 1 occurs (at the 0.1 and 0.2 branch).

Now we have the Huffman code for this image in Table 10-1.

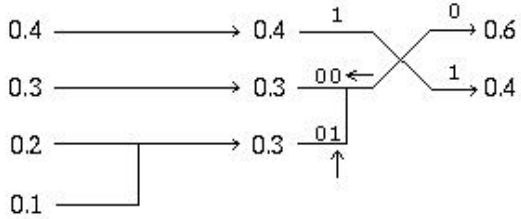
Figure 10.2-3: Huffman Coding Example, Step 5



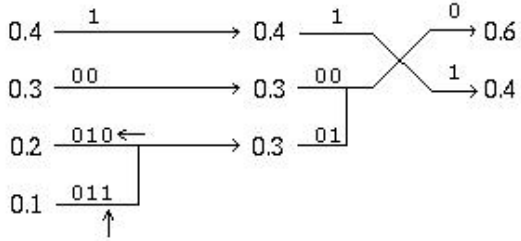
a) Assign 0 and 1 to the right-most probabilities



b) Bring 0 and 1 back along the tree



c) Append 0 and 1 to previously-added branches



d) Repeat the process until the original branch is labeled

TABLE 10-1

Original Gray Level (Natural Code)	Probability	Huffman code
g_0: 00_2	0.2	010_2
g_1: 01_2	0.3	00_2
g_2: 10_2	0.1	011_2
g_3: 11_2	0.4	1_2

EXAMPLE 10.2.4:

$$Entropy = -\sum_{i=0}^{3} p_i \log_2(p_i)$$

$$= -[(0.2)\log_2(0.2) + (0.3)\log_2(0.3) + (0.1)\log_2(0.1) + (0.4)\log_2(0.4)$$

$$\approx 1.846 \text{ bits/pixel}$$

(Note : $\log_2(x)$ can be found by taking $\log_{10}(x)$ and multiplying by 3.322)

$$L_{ave} = \sum_{i=0}^{L-1} l_i \; p_i$$

$$= 3(0.2) + 2(0.3) + 3(0.1) + 1(0.4)$$

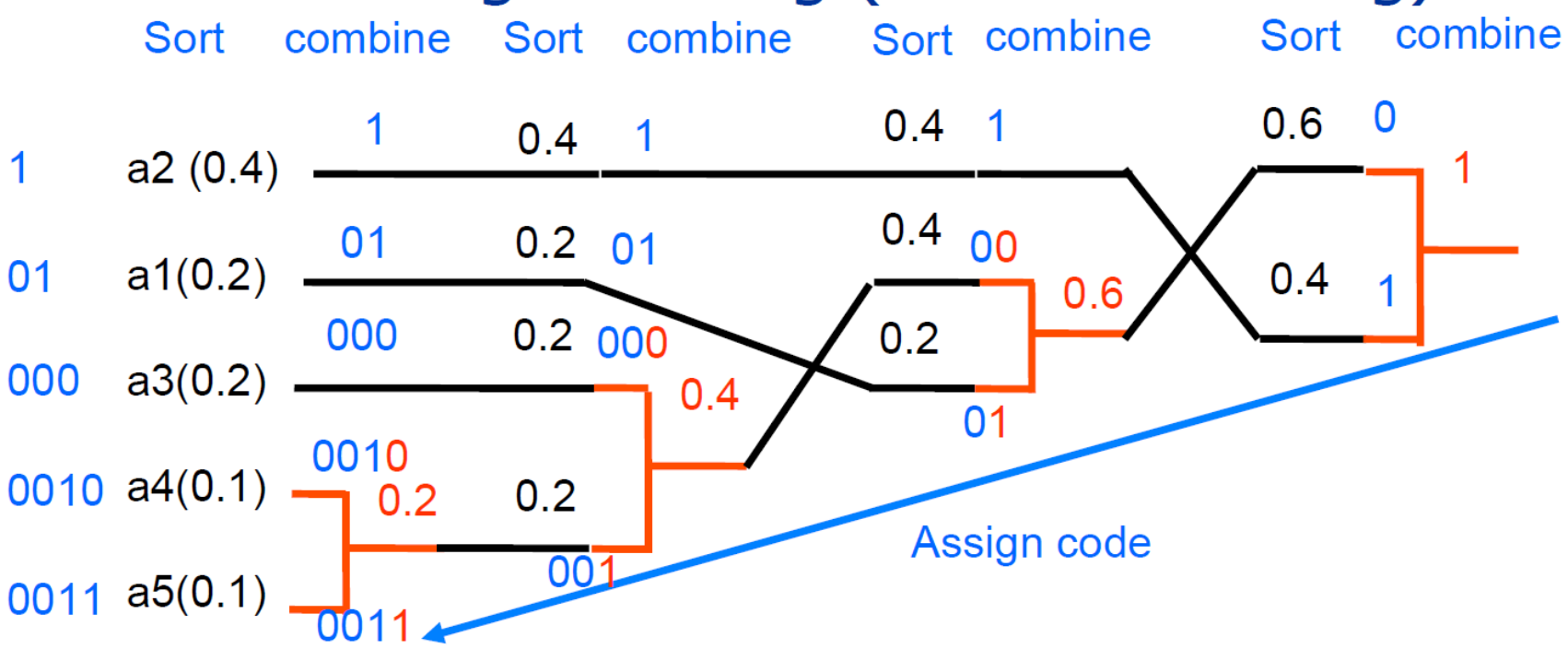
$$= 1.9 \text{ bits/pixel} \quad \text{(Average length with Huffman code)}$$

- In the example, we observe a 2.0 : 1.9 compression, which is about a 1.05 compression ratio, providing about 5% compression

- From the example we can see that the Huffman code is highly dependent on the histogram, so any preprocessing to simplify the histogram will help improve the compression ratio

Image Compression

Variable Length Coding (Huffman Coding)

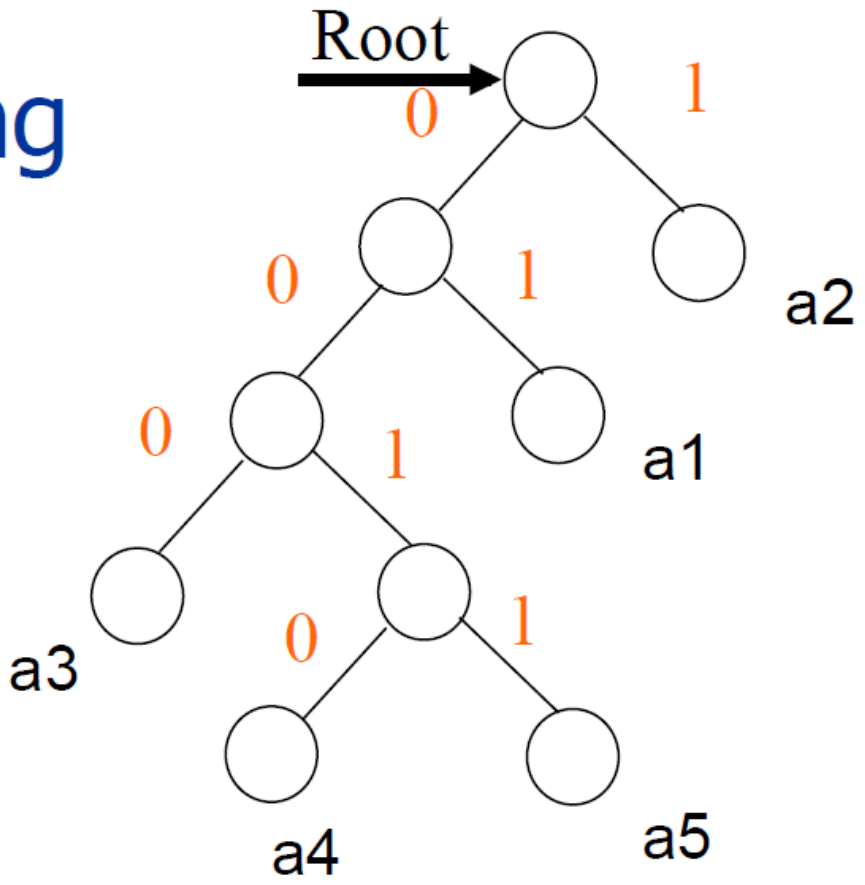


Huffman Decoding

Example:

00111010001

Decoding



Huffman Decoding

Example:

00111010001

a5

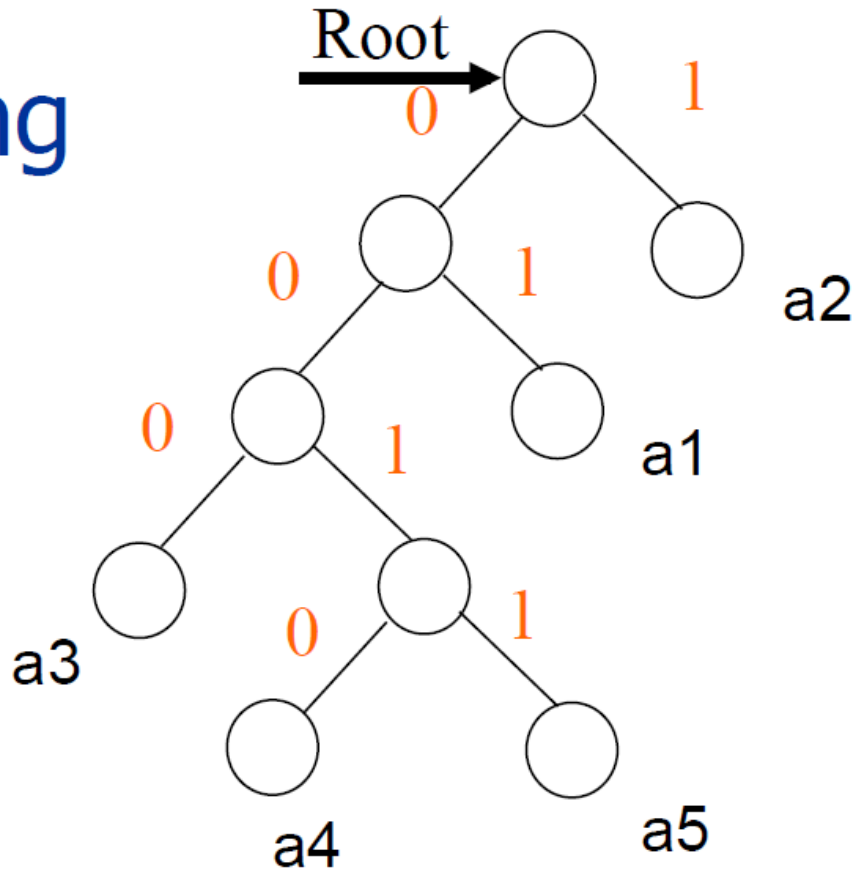
a2

a1

a3

a2

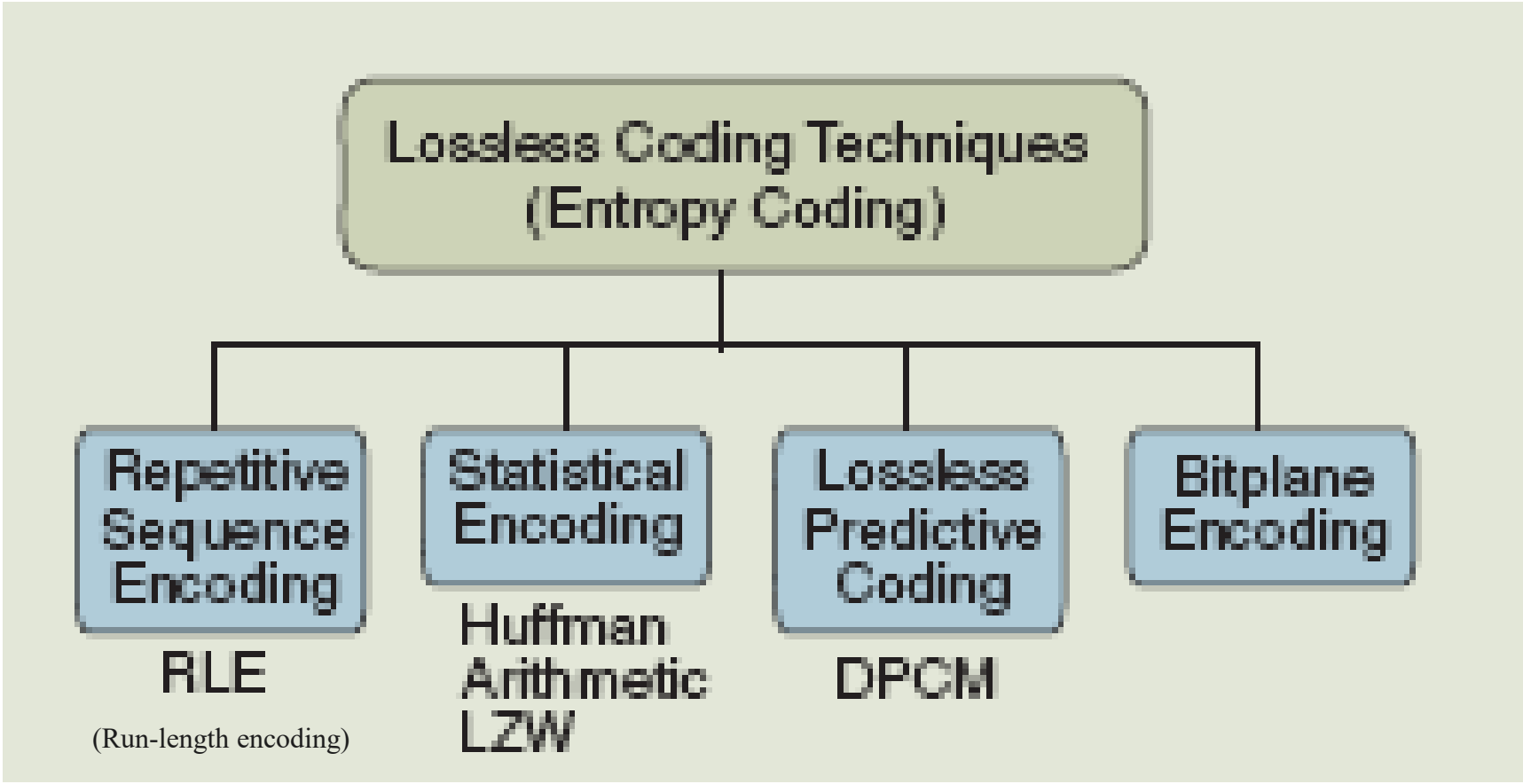
Decoding



- To decode the encoded data we require the Huffman tree. We iterate through the binary encoded data. To find character corresponding to current bits, we use the following simple steps:

- We start from the root and do the following until a leaf is found.
- If the current bit is 0, we move to the left node of the tree.
- If the bit is 1, we move to right node of the tree.
- If during the traversal, we encounter a leaf node, we print the character of that particular leaf node and then again continue the iteration of the encoded data starting from step 1.

Taxonomy of Lossless Methods



Arithmetic Coding

- Basic Idea:
- Like Huffman coding requires prior knowledge of probabilities
- Unlike Huffman coding, which assigns variable length codes to symbols arithmetic coding assigns codes to a variable group of symbols i.e. the message.
- There is no one-to-one correspondence between the symbol and its corresponding code word.
- The code word itself defines a real number within the half-open interval [0,1) and as more symbols are added, the interval is divided into smaller and smaller subintervals, based on the probabilities of the added symbols.

Arithmetic (or Range) Coding
(addresses coding redundancy)

- Huffman coding encodes source symbols **one** at a time which might not be efficient in general.
- Arithmetic coding assigns **sequences** of source symbols to **variable length** code words.

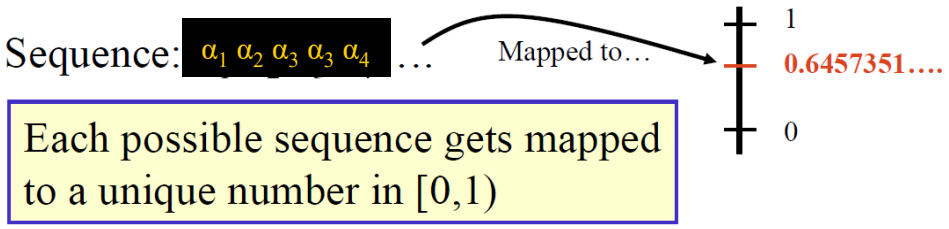
- **No** one-to-one correspondence:

  (source symbols $\not\leftrightarrow$ code words)

- **Slower** than Huffman coding but can achieve **higher** compression.

Arithmetic Coding – Main Idea

- Maps a **sequence** of symbols to a real number (**arithmetic code**) in the interval [0, 1).



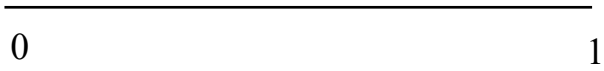
- The mapping is built **incrementally** (i.e., scanning source symbols in sequence) and depends on the source symbol **probabilities**.

Arithmetic Coding – Main Idea (cont'd)

Symbol sequence: $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$
known probabilities $P(\alpha_i)$

- Start with the interval [0, 1)

- A sub-interval of [0,1) is chosen to encode the first symbol α_1 in the sequence (**based on** $P(\alpha_1)$).



- A sub-interval **within** the previous sub-interval is chosen to encode the next symbol α_2 in the sequence (**based on** $P(\alpha_2)$).

- Eventually, the whole symbol sequence is encoded by choosing some number within the final sub-interval, e.g.:

final sub-interval arithmetic code

$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$ → [0.06752, 0.0688) → 0.068

Arithmetic Coding - Example

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

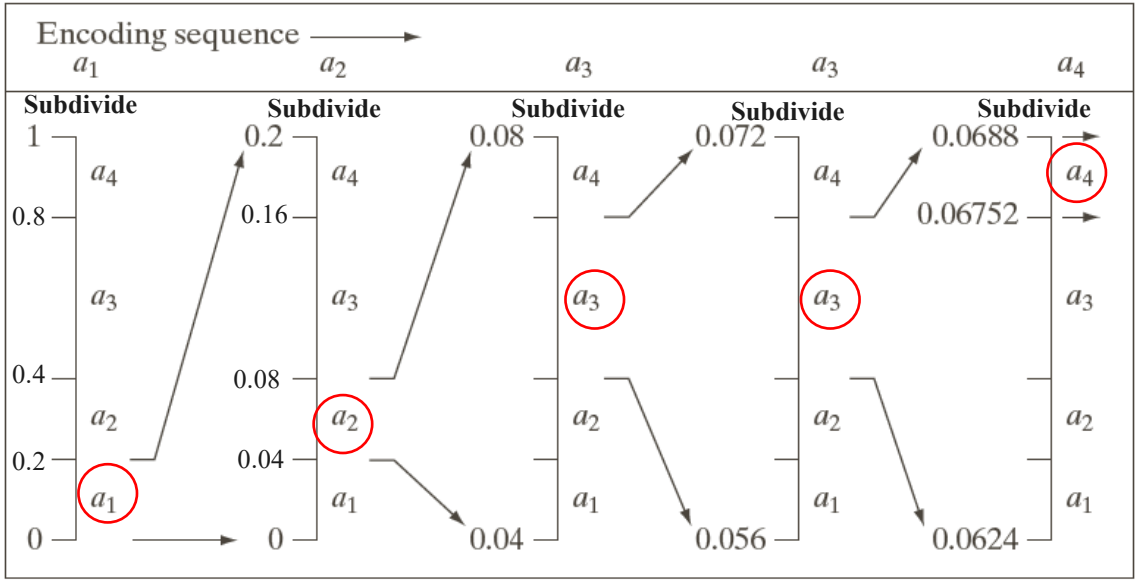
Subdivide [0,1) based on $P(\alpha_i)$

Encode

$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

[0.06752, 0.0688)
final sub-interval

arithmetic code: 0.068
(can choose any number within the final sub-interval)



Warning: finite precision arithmetic might cause problems due to truncations!

Arithmetic Coding - Example (cont'd)

- The arithmetic code 0.068 can be encoded using Binary Fractions:

  $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

  $0.0068 \approx 0.000100011$ (9 bits)

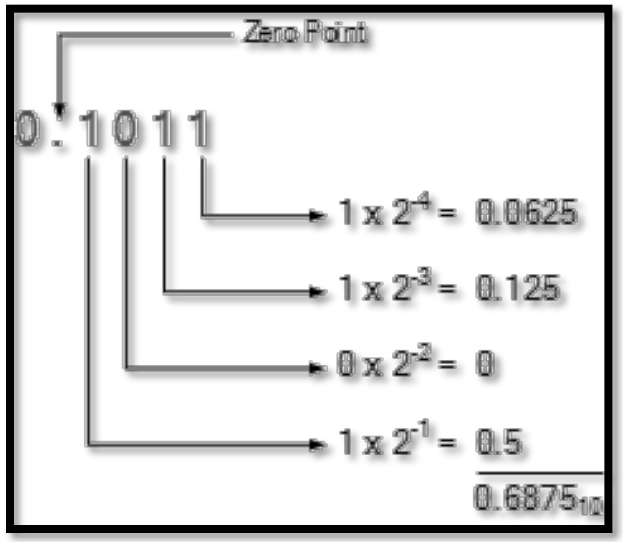
- Huffman Code:

  0100011001 (10 bits)

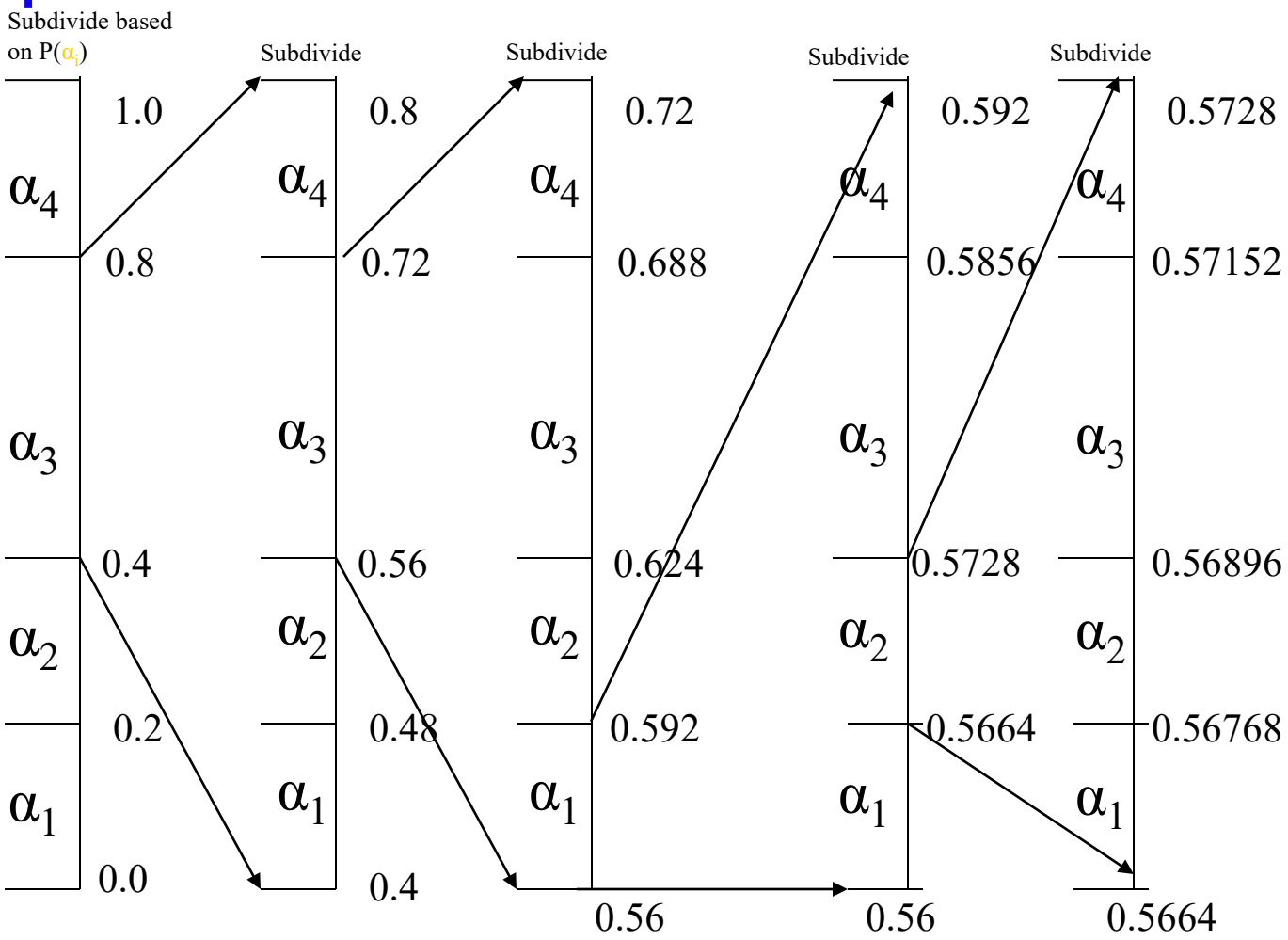
- Fixed Binary Code:

  5 x 8 bits/symbol = 40 bits

Example



Arithmetic Decoding - Example



Decode 0.572

$\alpha_3\ \alpha_3\ \alpha_1\ \alpha_2\ \alpha_4$

A special EOF symbol can be used to terminate iterations.

Run Length Coding

- Run-Length Encoding (RLE) is a basic data compression method that eliminates redundant information in a dataset by replacing consecutive repeated values with a count and the value itself. It works on various data types, including text, images, and numerical data.

- Follow the steps below to solve this problem:

1. Pick the first character from the source string.
2. Append the picked character to the destination string.
3. Count the number of subsequent occurrences of the picked character and append the count to the destination string.
4. Pick the next character and repeat steps 2, 3 and 4 if the end of the string is NOT reached.

Run: a string of the same symbol

Example

input: AAABBCCCCCCCCCAA
output: A3B2C9A2

compression ratio = 16/8 = 2

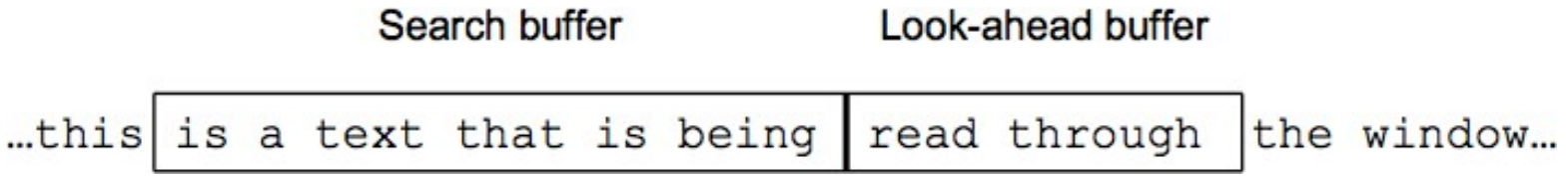
Dictionary Based Method

- Compressing multiple strings can be more efficient than compressing single symbols only (e.g. Huffman encoding).
- Strings of symbols are added to a dictionary. Later occurrences are referenced.
- Static dictionary: Entries are predefined and constant according to the application of the tex
- Adaptive dictionary: Entries are taken from the text itself and created on-the-fly

Dictionary Based Method – LZ77

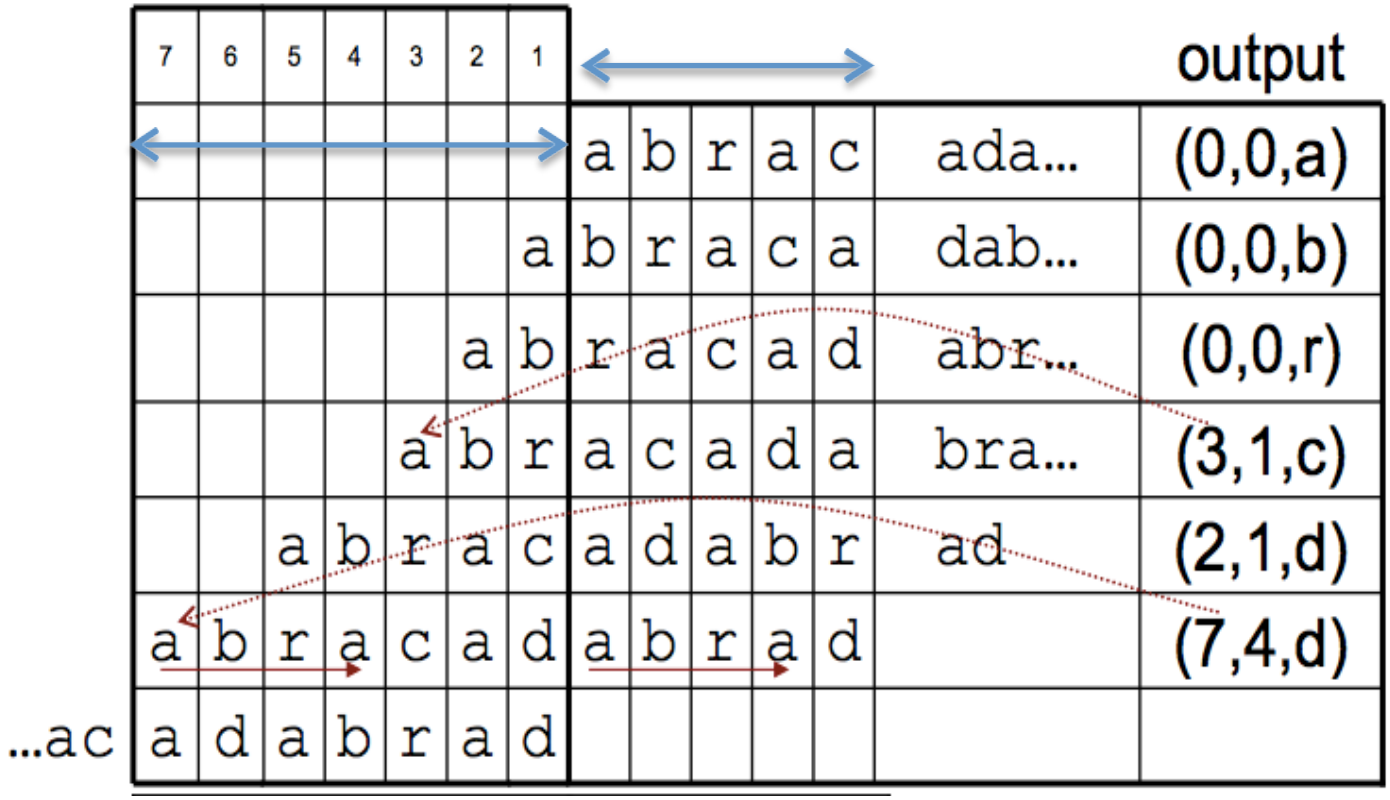
➢By Lempel and Ziv in 1977 about lossless compression with an adaptive dictionary.

- Runs through the text in a sliding window
- Two buffers are used - search (history) buffer and a look ahead buffer.
- The search buffer is used as dictionary
- Sizes of these buffers are parameters of the design



Encoding of the string:
abracadabrad

output tuple: (offset, length, symbol)



12 Characters are compressed into 6 tuples

Decoding

input		7	6	5	4	3	2	1
(0,0,a)								a
(0,0,b)							a	b
(0,0,r)						a	b	r
(3,1,c)				a	b	r	a	c
(2,1,d)		a	b	r	a	c	a	d
(7,4,d)	abrac	a	d	a	b	r	a	d

Dictionary Based Method – LZW

- Extended by Welch (Lempel, Ziv and Welch)

- This coding scheme has been adopted in a variety of imaging file formats, such as the graphic interchange format (GIF), tagged image file format (TIFF) and the portable document format (PDF).

Dictionary Based Method – LZW

- Extended by Welch (Lempel, Ziv and Welch)

- Unlike Huffman coding and arithmetic coding, this coding scheme does not require a priori knowledge of the probabilities of the source symbols.
- The coding is based on a “dictionary” or “codebook” containing the source symbols to be encoded. The coding starts with an initial dictionary, which is enlarged with the arrival of new symbol sequences.
- There is no need to transmit the dictionary from the encoder to the decoder. The decoder builds an identical dictionary during the decoding process

Example: 32 32 34 32 34 32 32 33 32 32 32 34

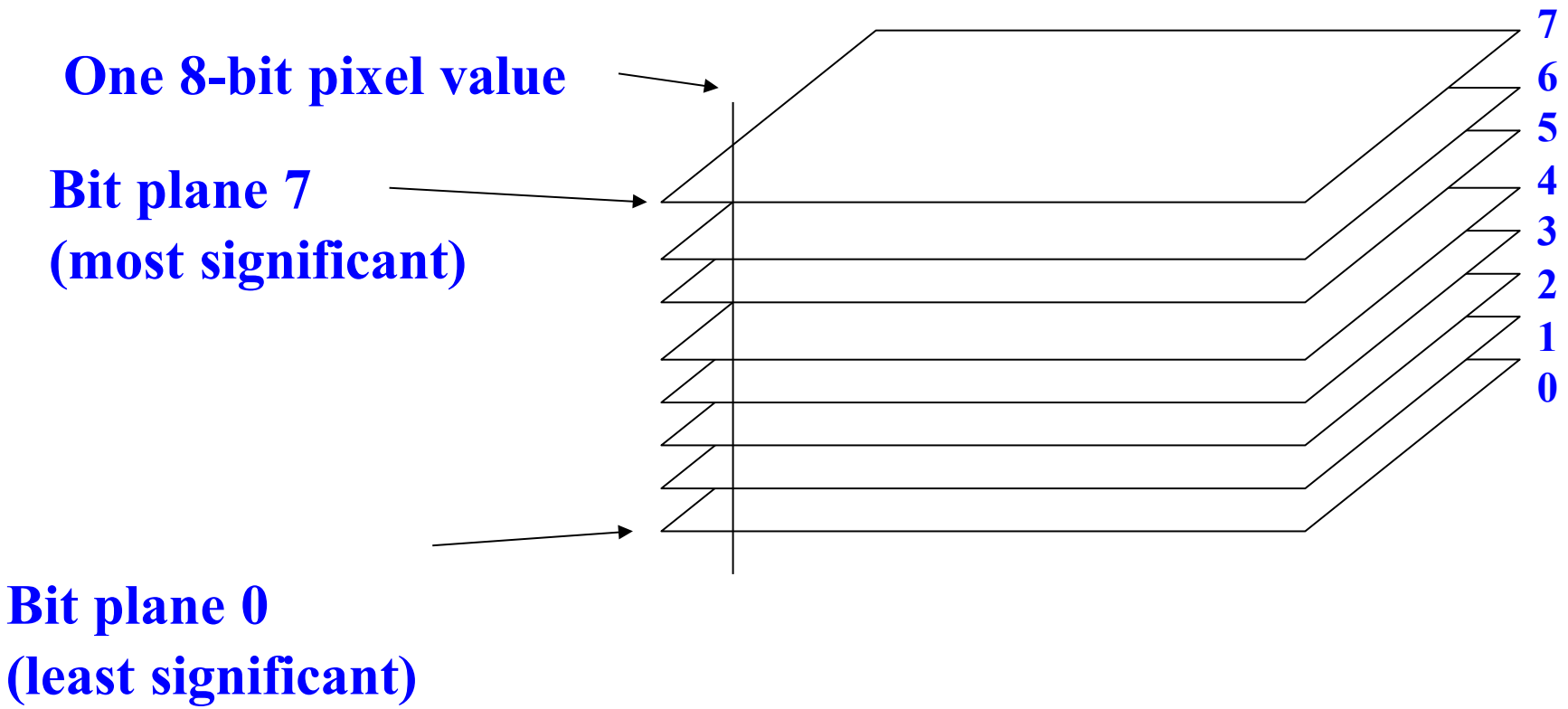
Consider a dictionary of size 256 locations (numbered 0 to 255) that contains entries corresponding to each pixel intensity value in the range 0-255.

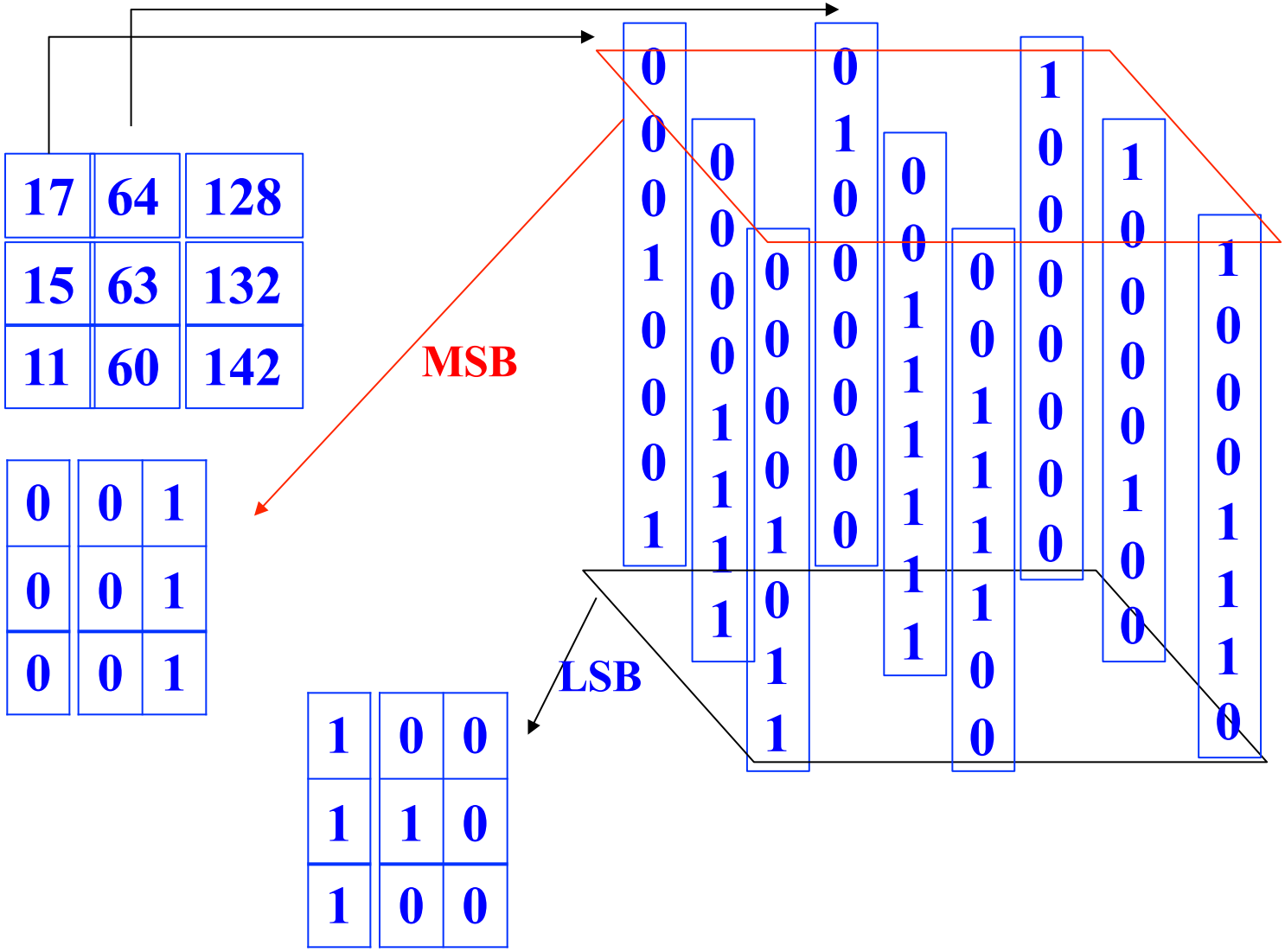
Currently Recognized Sequence	Pixel being processed	Encoded Output	Dictionary Location (Code word)	Dictionary Entry
	32			
32	32	32	256	32-32
32	34	32	257	32-34
34	32	34	258	34-32
32	34			
32-34	32	257	259	32-34-32
32	32			
32-32	33	256	260	32-33
33	32	33	261	33-32
32	32			
32-32	32	256	262	32-32-32
32	34			
32-34		257		

Bit Plane Coding

- This method is based on decomposing a multilevel (monochrome or color) image into a series of binary images and compressing each binary image.

- We review different approaches for decomposition and compression of binary images.


One 8-bit pixel value
Bit plane 7
(most significant)
Bit plane 0
(least significant)
7
6
5
4
3
2
1
0


17 64 128
15 63 132
11 60 142
MSB
LSB
0 0 1
0 0 1
0 0 1
1 0 0
1 1 0
1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 1 1 1 1
0 0 0 0 0 0 1 1
0 1 0 0 0 0 0 0
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 0
1 0 0 0 0 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 1 1 1 0

Example

- Let us consider a simple 4x4 array, whose elements are integers in the range 0-15 . The corresponding 4-bit binary representations are indicated within the Parenthesis's shown below:

13(1101)	12(1100)	13(1101)	9(1001)
15(1111)	13(1101)	11(1011)	10(1010)
14(1110)	11(1011)	12(1100)	9(1001)
11(1011)	12(1100)	14(1110)	7(0111)

The corresponding 4 bit planes are shown below:

1 1 1 1	1 1 1 0	0 0 0 0	1 0 1 1
1 1 1 1	1 1 0 0	1 0 1 1	1 1 1 0
1 1 1 1	1 0 1 0	1 1 0 0	0 1 0 1
1 1 1 0	0 1 1 1	1 0 1 1	1 0 0 1
Bit-plane-3(MSB)	Bit-plane-2	Bit-plane-1	Bit-plane-0(LSB)

Predictive Coding

Basic premise: Current pixel is similar to the previous pixel (coherence)

Differential Coding

$$d(x,y) = I(x,y) - I(x-1,y)$$

d(x,y) prediction error which is to be encoded.

- Provides a data representation where code words express source symbol deviations from predicted values (usually values of neighboring pixels).
- Predictive coding efficiently reduces interpixel redundancies
  - 1D & 2D – pixels are predicted from neighboring pixels
  - 3D – pixels are predicted between frames as well

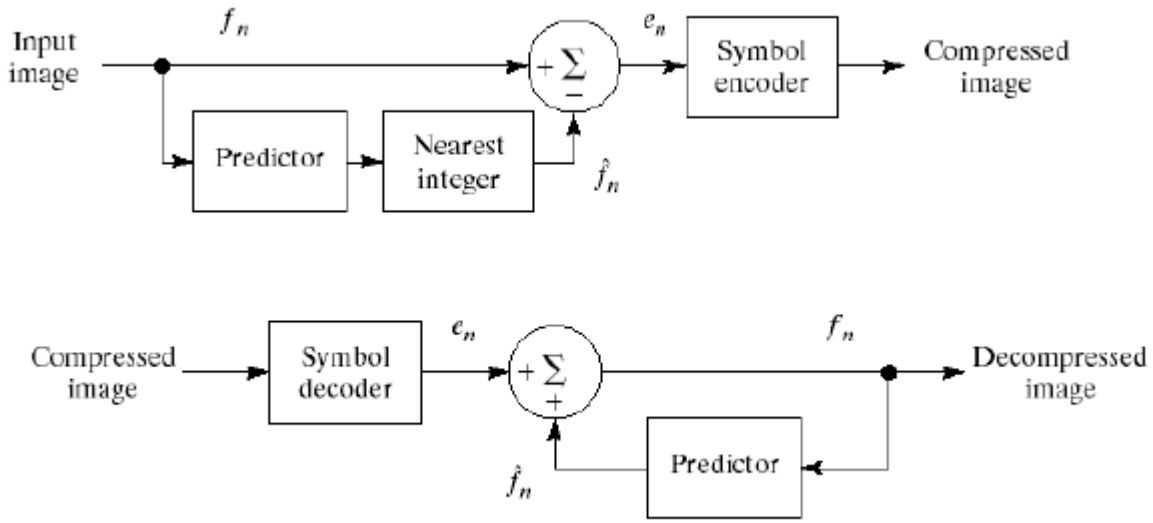
- Works well for all images with a high degree of interpixel redundancies. Works in the presence of noise (just not as efficiently)
- Input data
  222222222266666666666669999999999999999
- Code
  2000000000040000000000003000000000000000

Predictive coding can be used in both lossless and lossy compression schemes

Lossless Predictive Coding

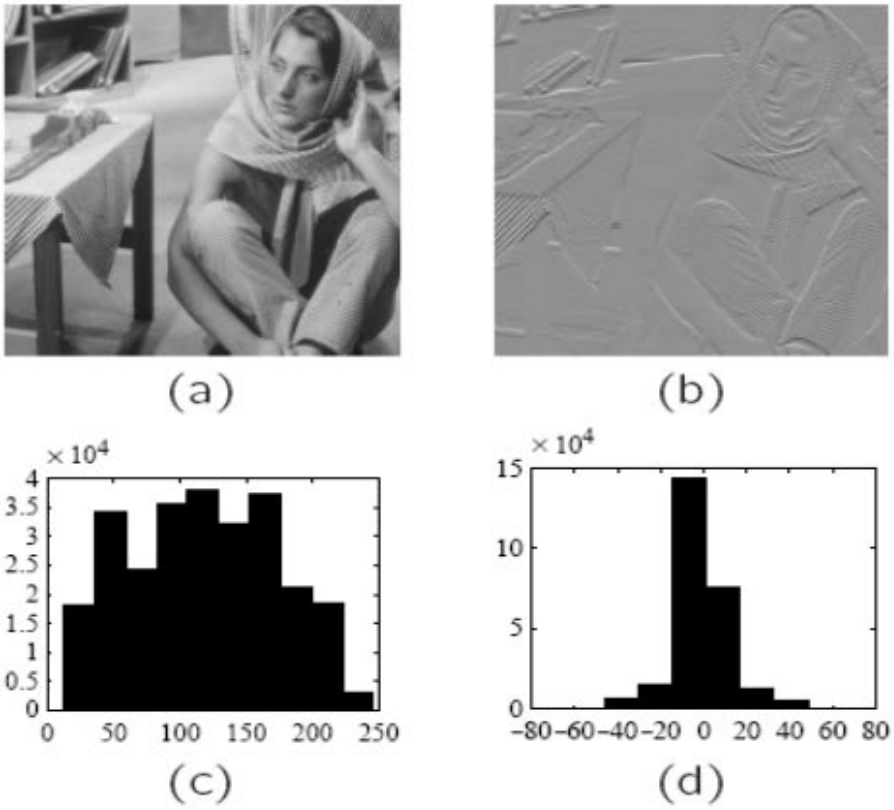
- In this technique there is no need to decompose the image into bit planes.
- This technique eliminates the interpixel redundancies of closely spaced pixels.
- How? By extracting and coding only the new information in each pixel
- New information: the difference between actual and predicted value of that pixel

- Predictors in encoder and decoder are the same.
- Various local, global and adaptive methods can be used to generate the prediction



- Linear predictor is common:
- Previous pixels are used to estimate the value of the current pixel
- The previous pixels could be on the same row (column) with the current pixel (1-D prediction) or around the current pixel (2-D)

$$\hat{f}(n) = round\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$



Distributions for Original versus Derivative Images. (a,b): Original gray-level image and its partial derivative image; (c,d): Histograms for original and derivative images.

Lossy Image Compression

- Psychovisual Redundancy
- Keep more important information
- Trade off between loss (degradation) and compression

Original

Compression Ratio: 7.7

Compression Ratio: 12.3

Compression Ratio: 33.9

Compression Ratio 33.9

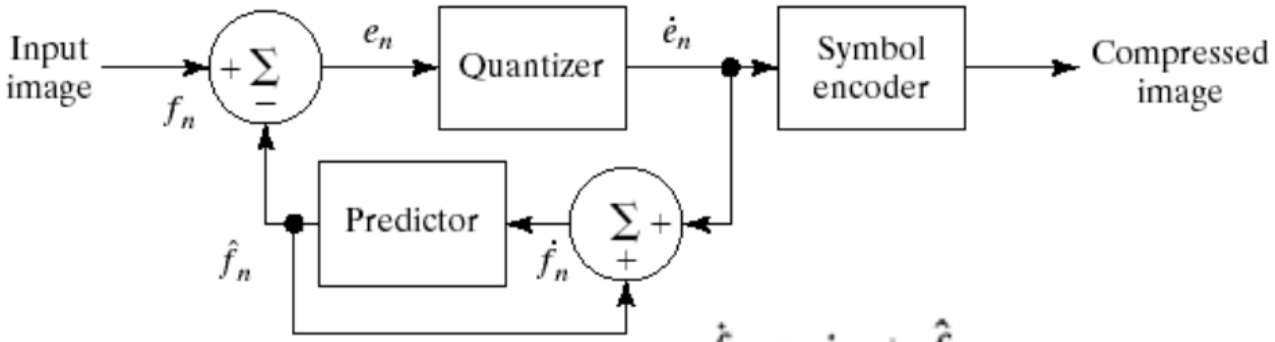
Original

Compression Ratio 7.7

Compression Ratio 33.9

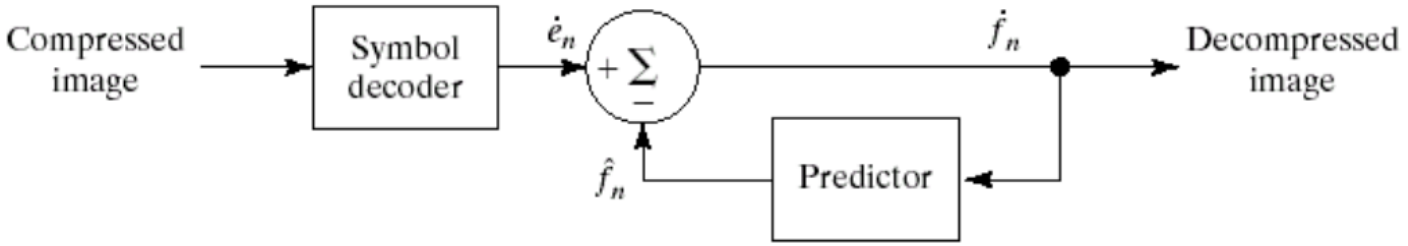
Predictive Coding: Lossy

Compression



Decompression

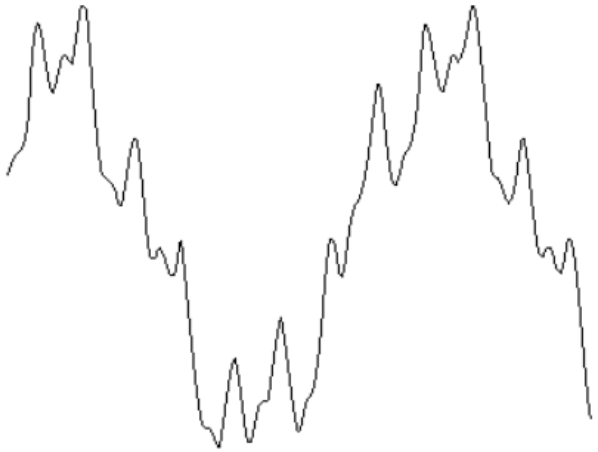
$$\dot{f}_n = \dot{e}_n + \hat{f}_n$$

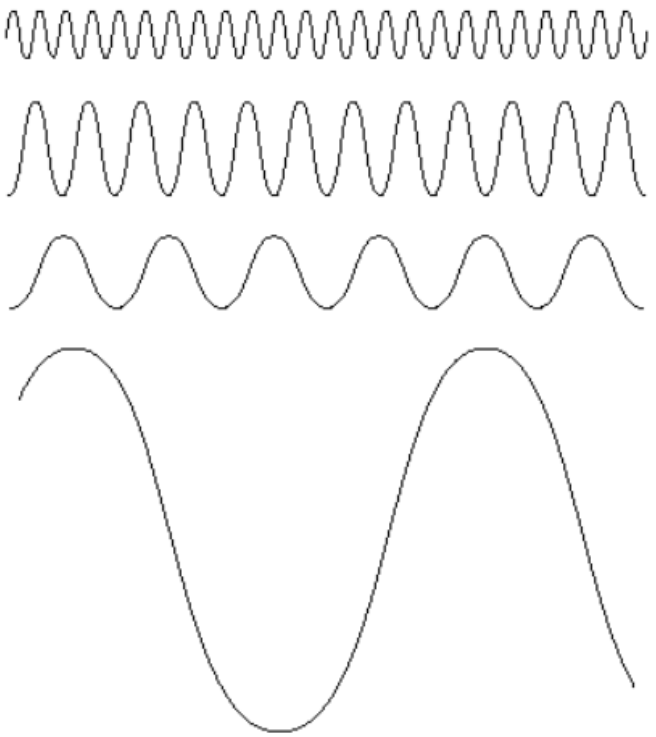


Transform Coding

- Transformation
  - Represent information in anther space
  - Identify and remove correlation
  - Quantization of transform coefficients

    Information loss!
  - Example
  - time/space → frequency (Fourier Transform)
  - Inverse transformation
    - Bring back information in the original space





- A function as sum of sines and cosines

Fourier Transform

Mathematically

Forward

Inverse

1-D:

$$F(u) \equiv \Im\{f(x)\} = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$

$$f(x) \equiv \Im^{-1}\{F(u)\} = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

2-D:

$$F(u,v) = \iint f(x,y) e^{-j2\pi(ux+vy)} dx\, dy$$

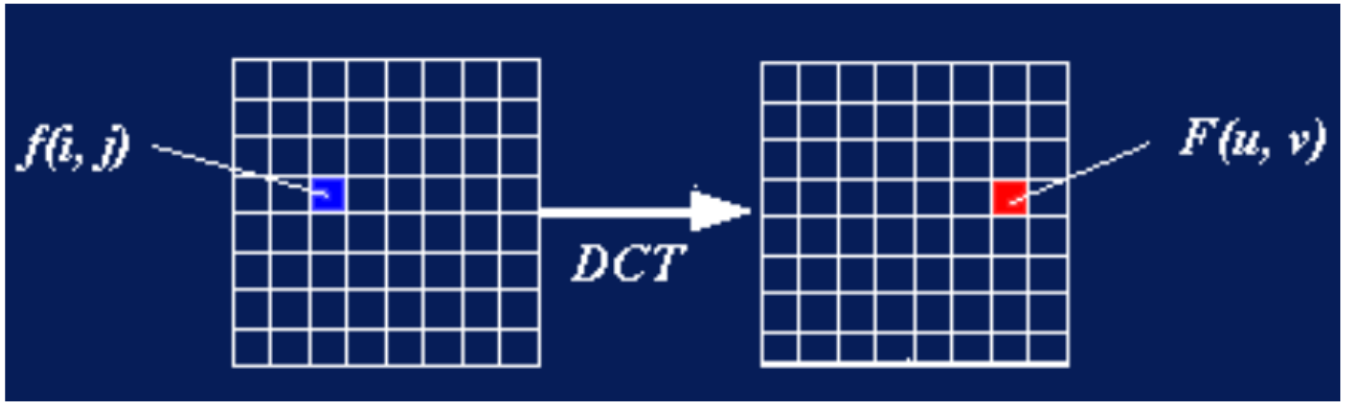
$$f(x,y) = \iint F(u,v) e^{j2\pi(ux+vy)} du\, dv$$

Discrete Fourier Transform

Fast Fourier Transform (FFT)

Discrete Cosine Transform

Popular transform for its performance and efficiency



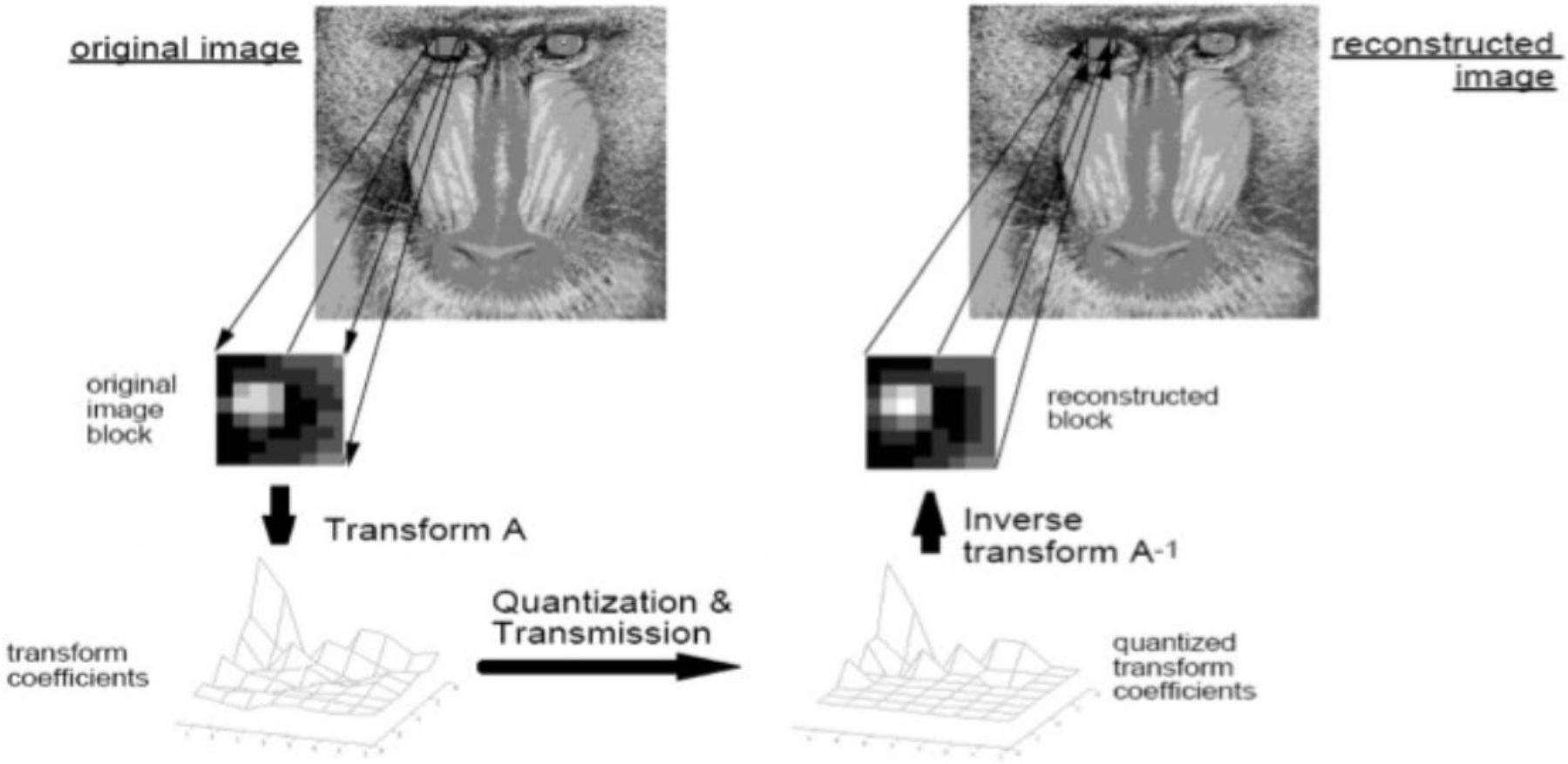
- Discrete Cosine Transform

Forward transform

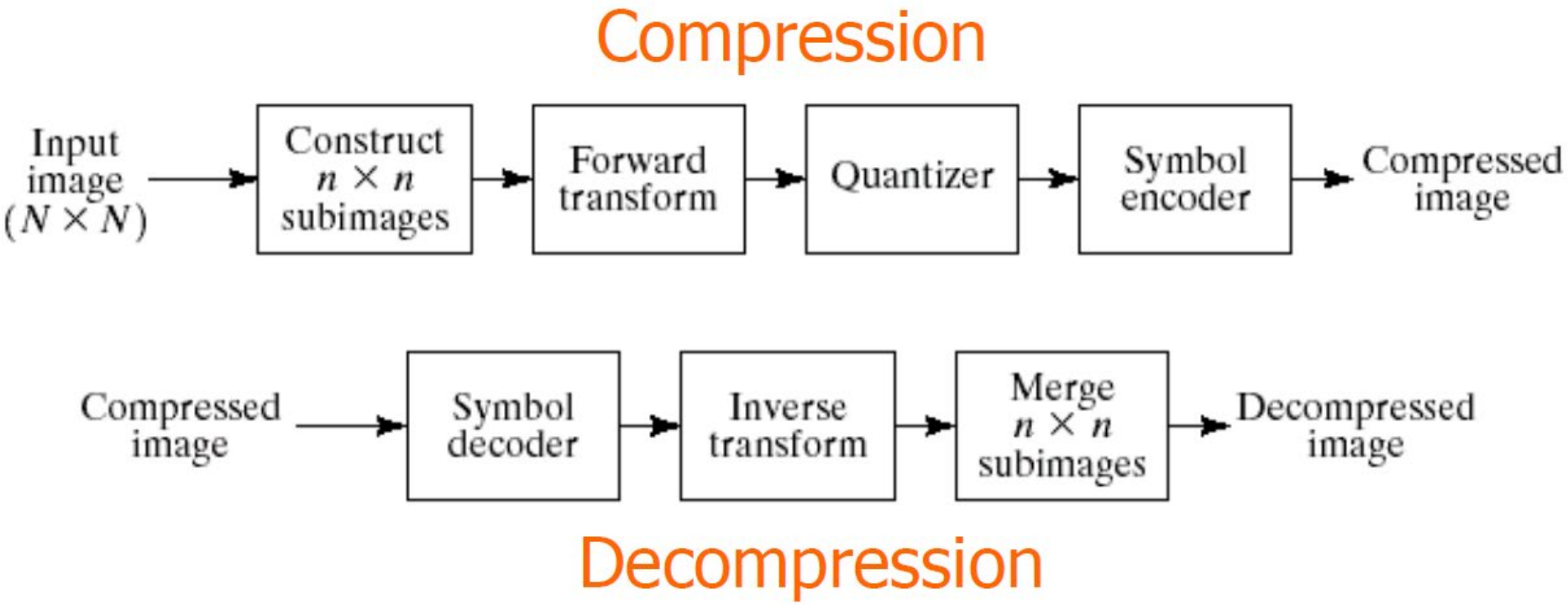
$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(x,y) \cos[\frac{(2x+1)u\pi}{2N}] \cos[\frac{(2y+1)v\pi}{2N}]$$

Inverse transform

$$f(x,y) = \frac{2}{N} \sum_{v=0}^{N-1} \sum_{u=0}^{N-1} C(u)C(v)F(u,v) \cos[\frac{(2x+1)u\pi}{2N}] \cos[\frac{(2y+1)v\pi}{2N}]$$

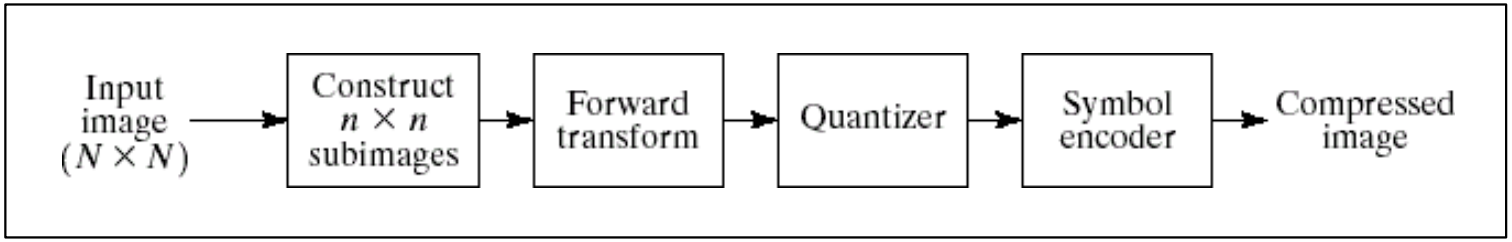

original image
reconstructed image
original image block
reconstructed block
Transform A
Inverse transform A-1
Quantization & Transmission
transform coefficients
quantized transform coefficients

Transform Coding Pipeline



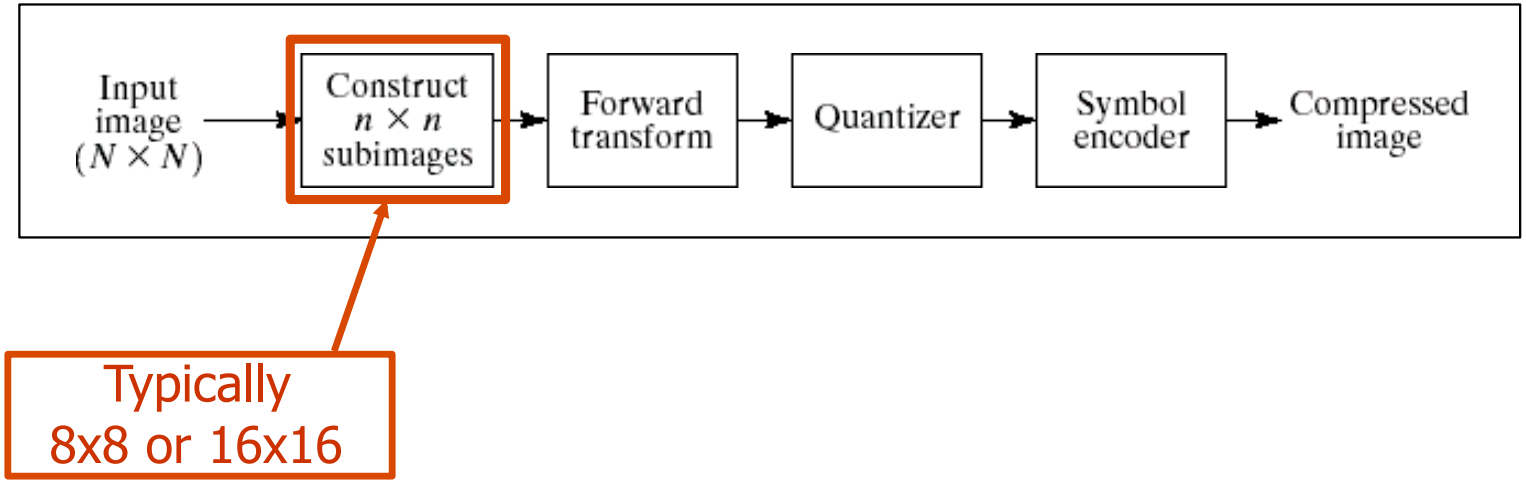
Transform Coding
Pipeline

Compression



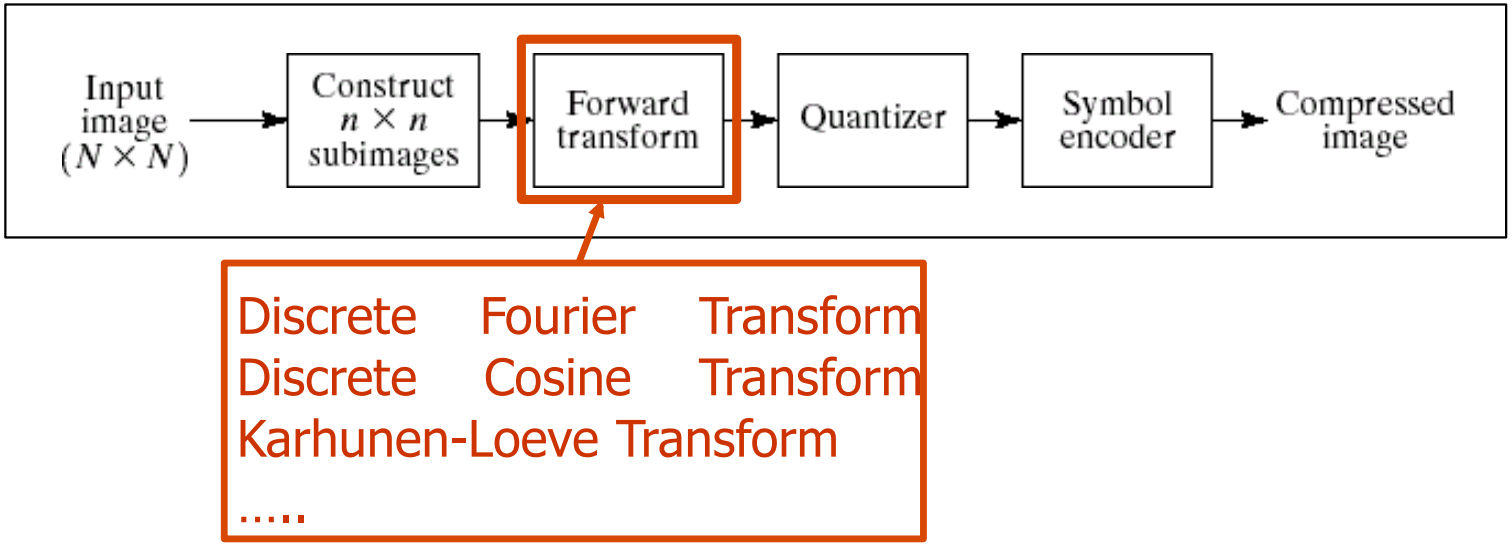
Transform Coding
Pipeline

Compression

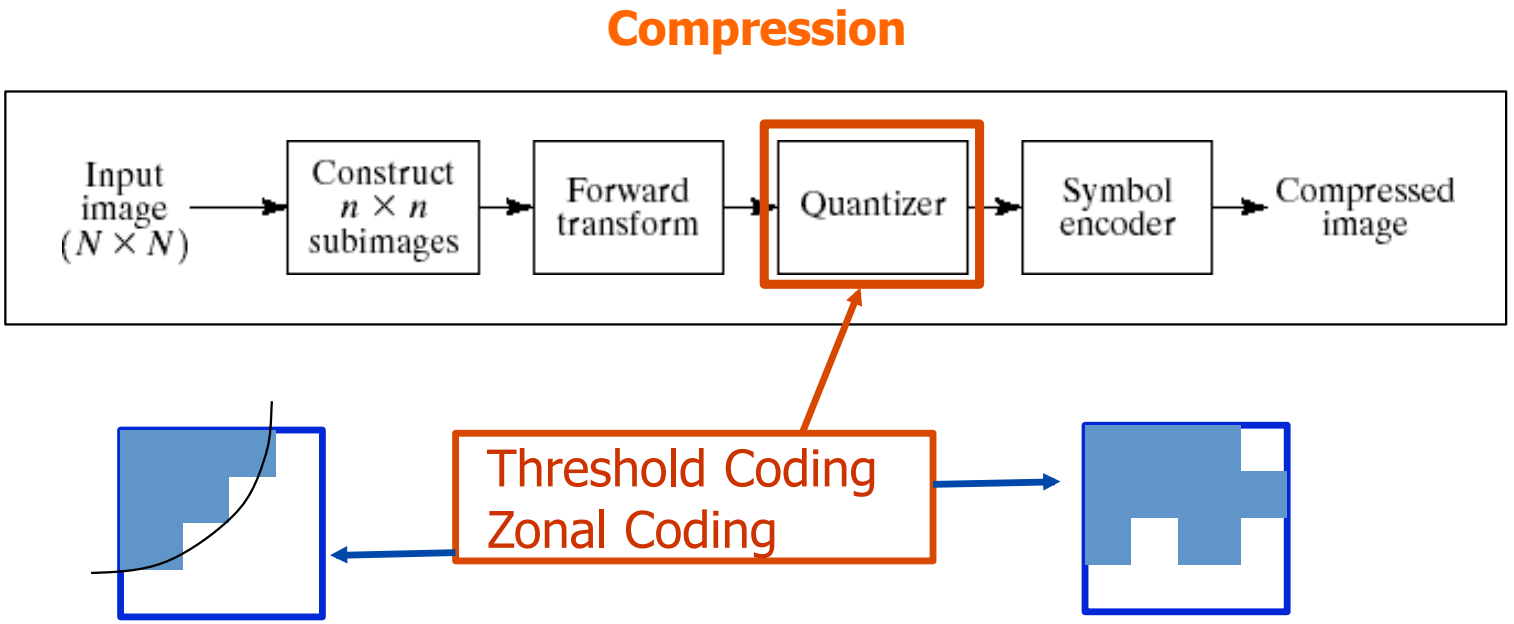


Transform Coding
Pipeline

Compression

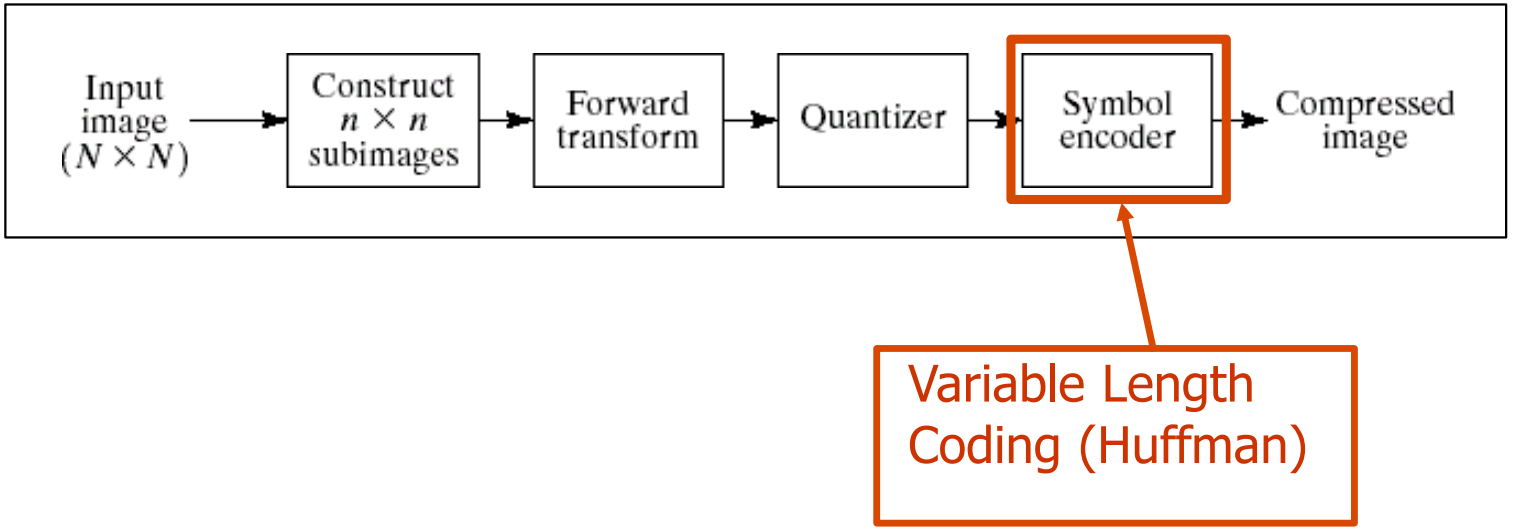


Transform Coding
Pipeline



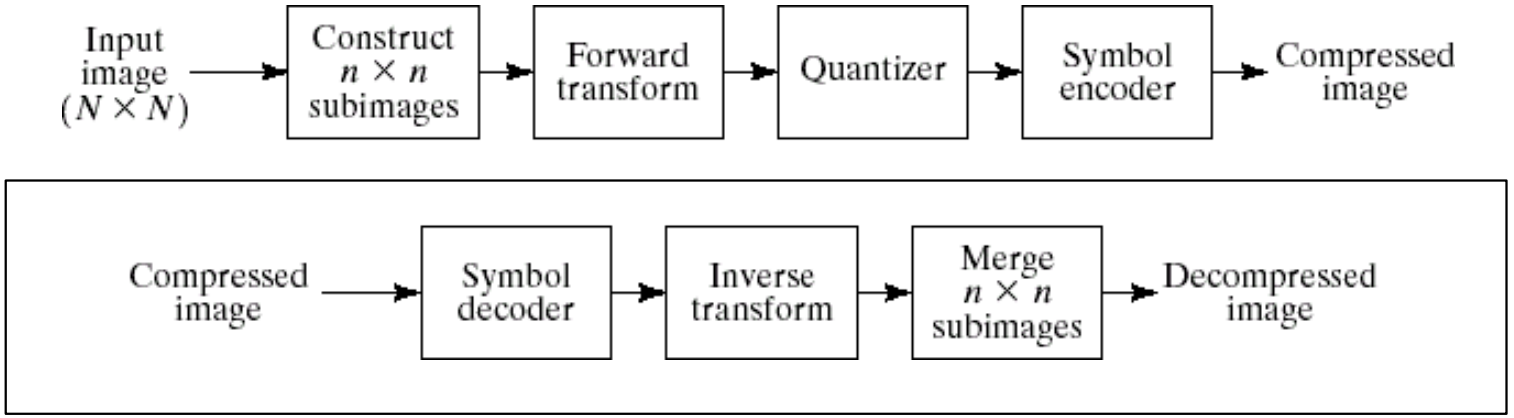
Transform Coding
Pipeline

Compression



Transform Coding
Pipeline

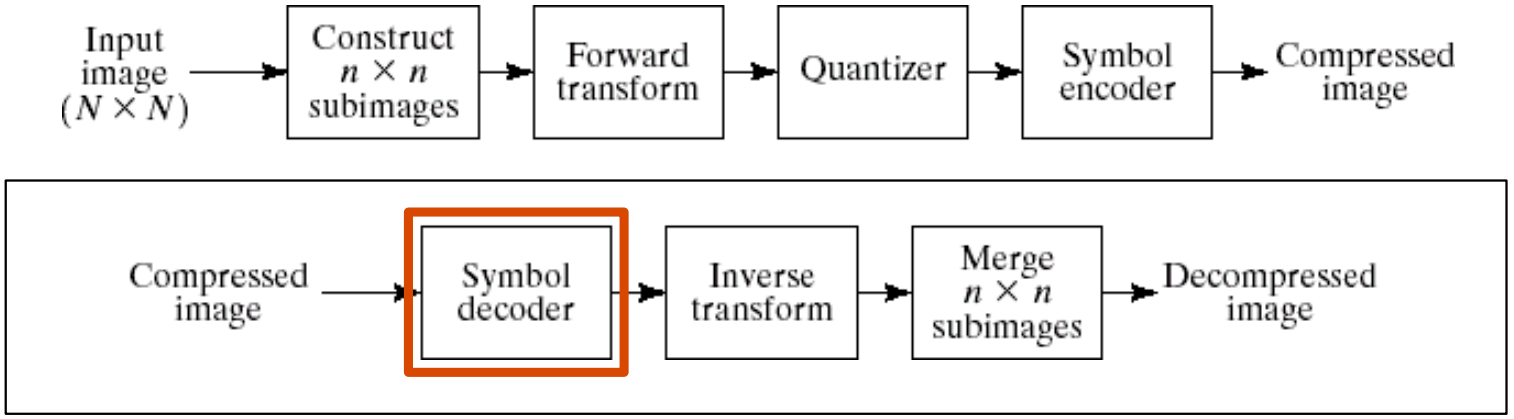
Compression



Decompression

Transform Coding
Pipeline

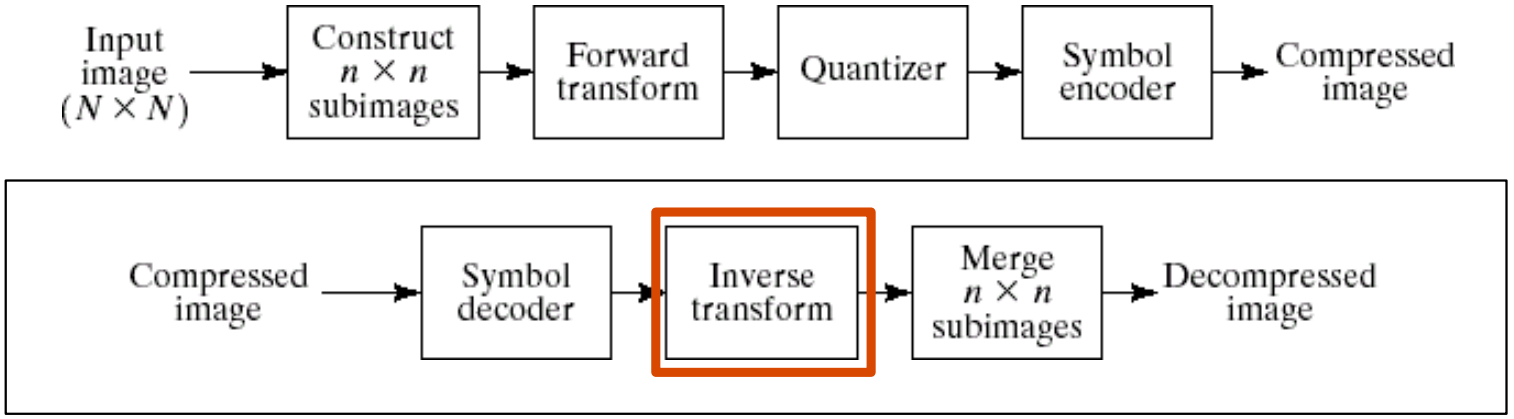
Compression



Decompression

Transform Coding
Pipeline

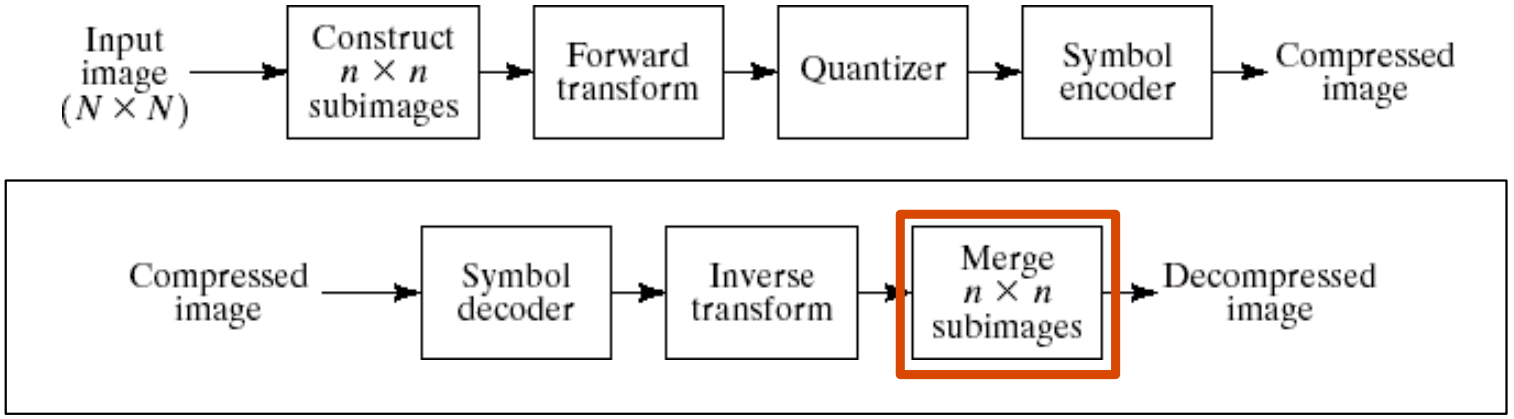
Compression



Decompression

Transform Coding
Pipeline

Compression



Decompression

JPEG Compression

- JPEG Standard
- JPEG - Joint Photographic Experts Group

- Compression of generic continuous-tone still image
- International standard in 1992
- Typical compression 10:1 to 50:1
- Allow for lossy and lossless compression
  - DCT-based lossy compression Predictive-based lossless compression

JPEG Compression

JPEG Standard

JPEG Compression

JPEG Standard

Number of modes of operation

Sequential: encoded in left-to-right, top-to-bottom scan
Progressive: encoded in multiple scans to first produce a quick, rough decoded image when the transmission time is long
Hierarchical: encoded at multiple resolution to allow accessing low resolution without full decompression
Lossless : decompressed image is identical to original

JPEG Compression

JPEG Standard

Baseline - Sequential

- Simple, lossy compression
  - DCT-based transform coding pipeline
- Preparation
  - Shift to zero-mean by subtracting 128 ➔ [-128, 127]
  - Color space transformation (YCbCr/YUV)
  - Down sampling of color components

JPEG Compression

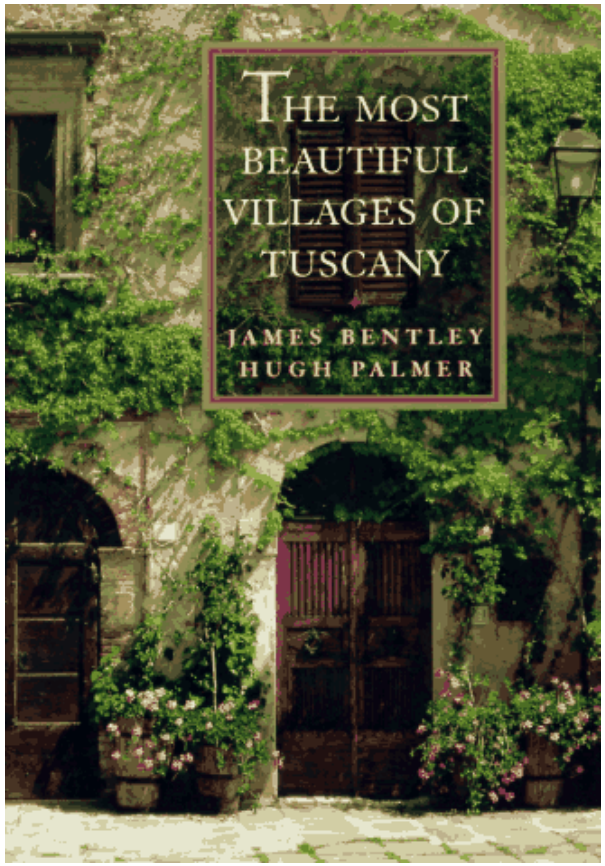
JPEG Standard: Steps

Color space transformation

$Y = 0.3\ R + 0.6\ G + 0.1\ B$ $Cb = 0.5\ (B-Y) + 0.5$

$Cr = (1/1.6)\ (R-Y) + 0.5$

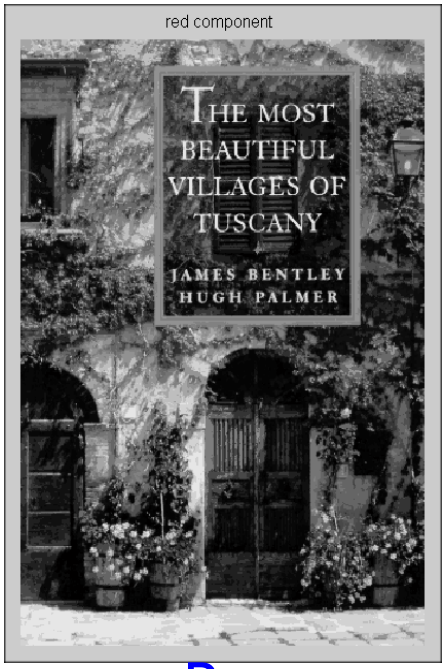
Color components can have lower spatial resolution than luminance



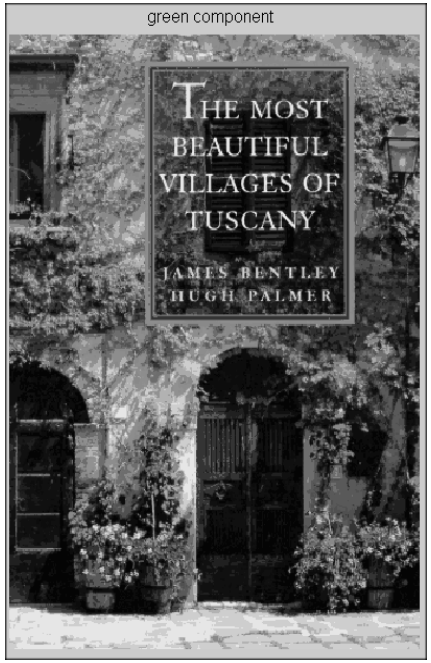
JPEG Compression

JPEG Standard: Steps

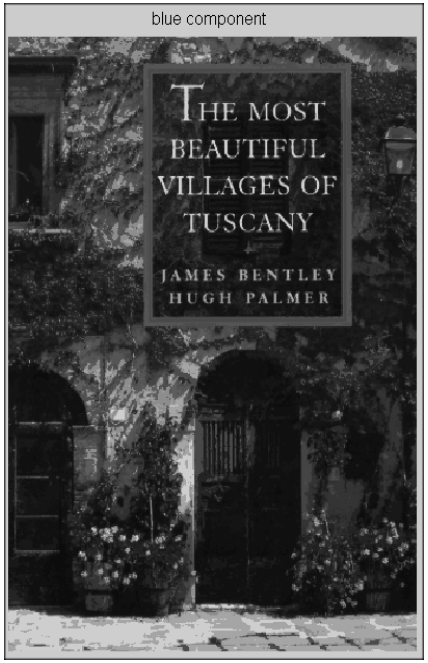
Color space transformation



R



G

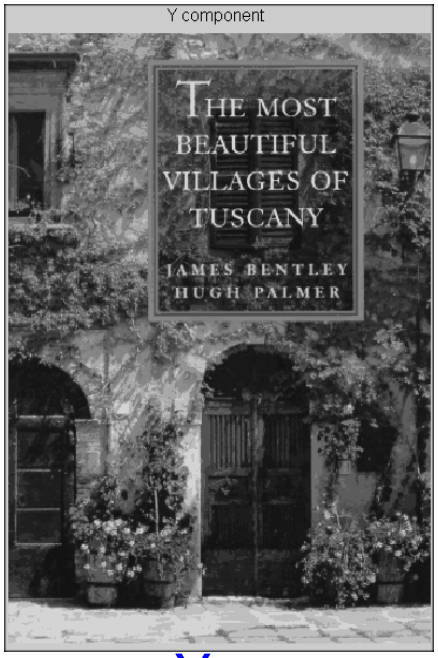


B

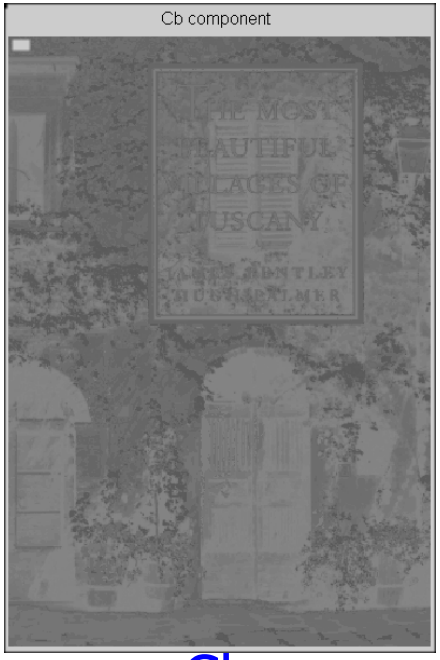
JPEG Compression

JPEG Standard: Steps

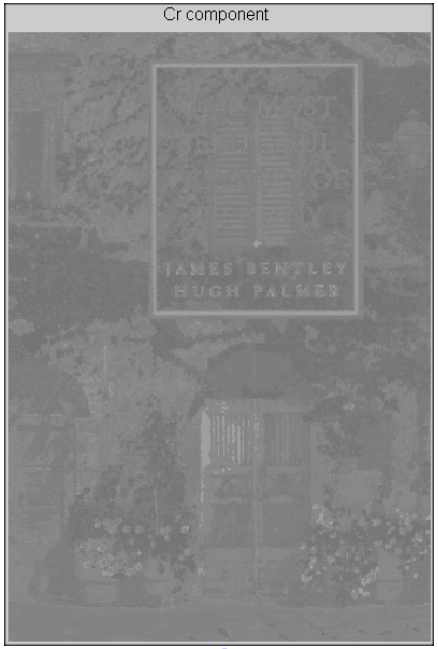
Color space transformation



Y



Cb



Cr

JPEG Compression

JPEG Standard: Steps

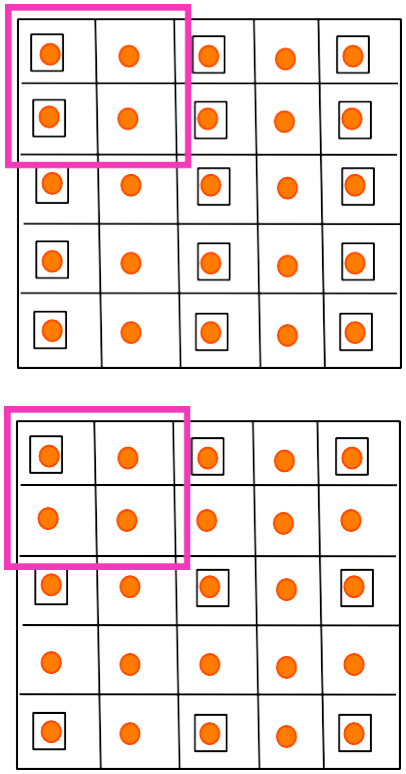
Down sampling

- 4-2-2

Y

Cb/Cr

- 4-1-1



JPEG Compression

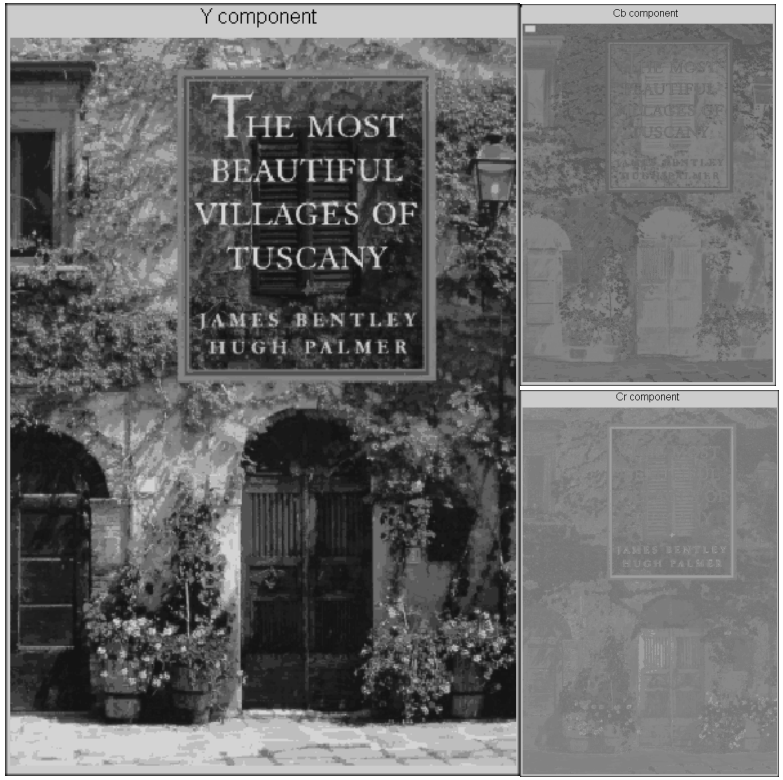
JPEG Standard: Steps

Down sampling

Example

4-1-1

Y



Cb

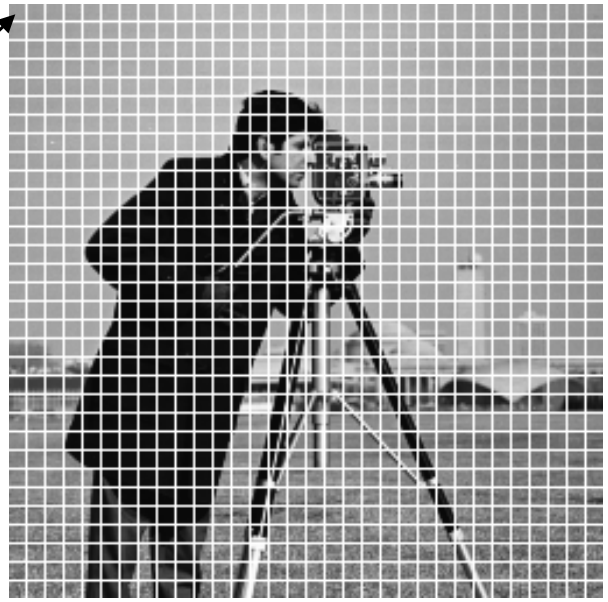
Cr

JPEG Compression

JPEG Standard: Steps

Divide image in to sub images

Subimage
of size 8x8

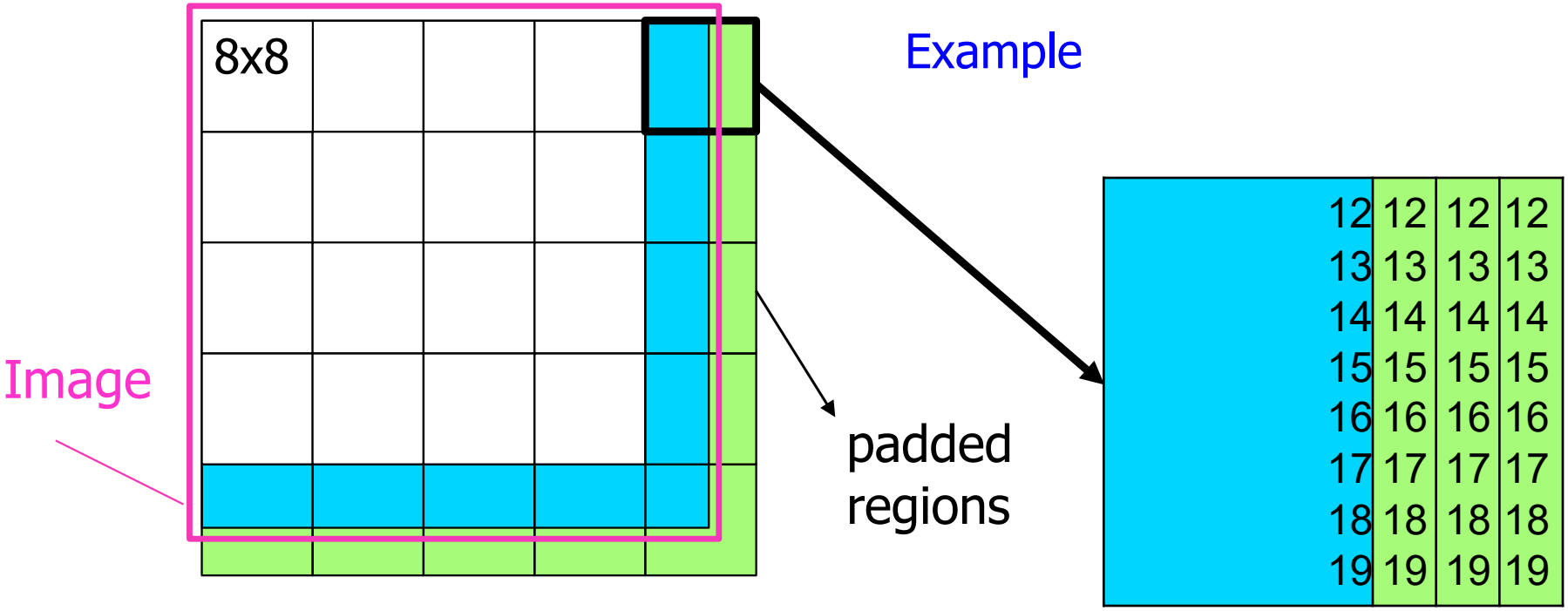


What if the image size is not a multiple of 8x8 block?

JPEG Compression

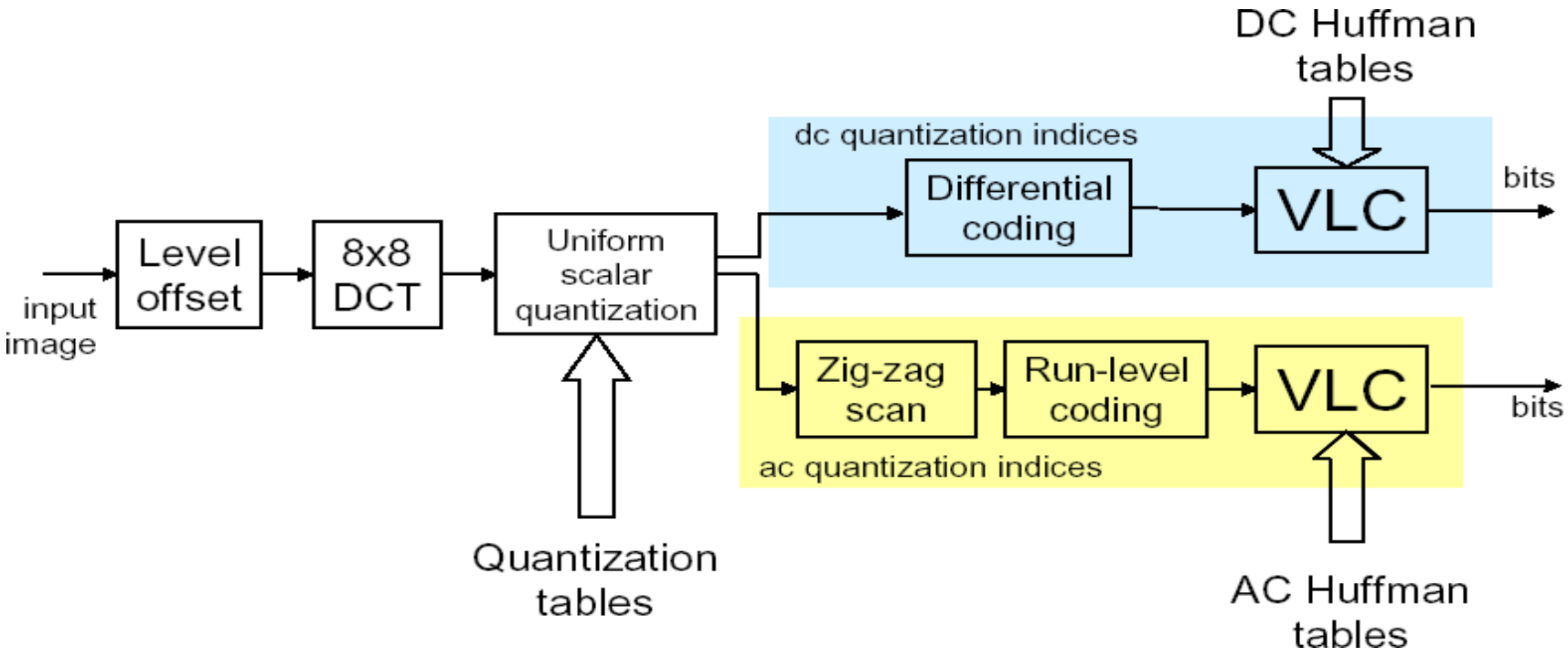
JPEG Standard: Steps

Divide image in to sub images



JPEG Compression

JPEG Standard: Steps



JPEG Compression

JPEG Standard: Steps

Example

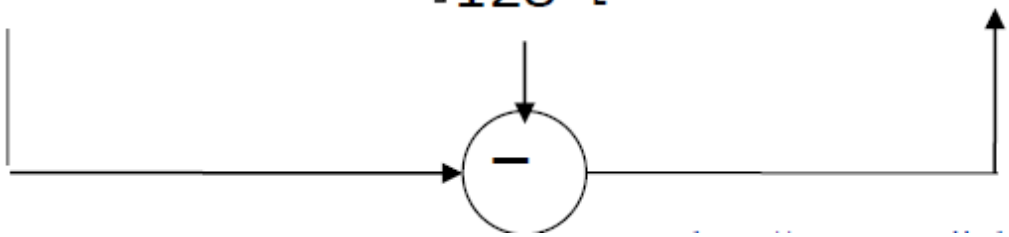
Image

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

JPEG Compression

Preparation Shift to zero-mean by subtracting 128

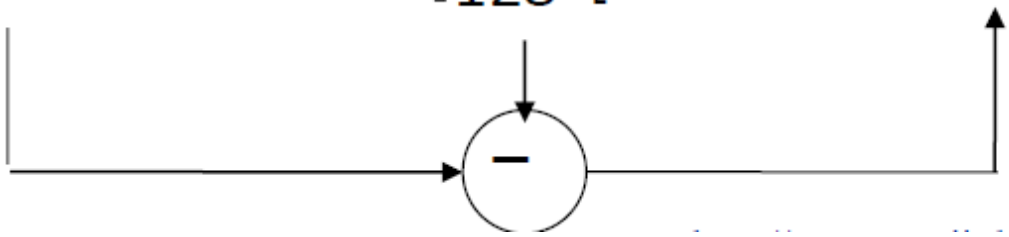
$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix} - 128 = \begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$



JPEG Compression

Preparation Shift to zero-mean by subtracting 128

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix} - 128 = \begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$



JPEG Compression

Forward DCT

$$\begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

8·8 DCT

$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}$$

JPEG Compression

Quantization

Q-table : specifies quantization stepsize

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

JPEG Compression

Quantization

Q-table : specifies quantization stepsize

- Q-table can be specified by user
- Q-table is scaled up/down by a chosen quality factor
- Quantization stepsize Q_{ij} is dependent on the coordinates (i,j) within the 8-by-8 block
- Quantization stepsize Q_{ij} increases from top-left to bottom-right

JPEG Compression

Quantization

x_{ij}

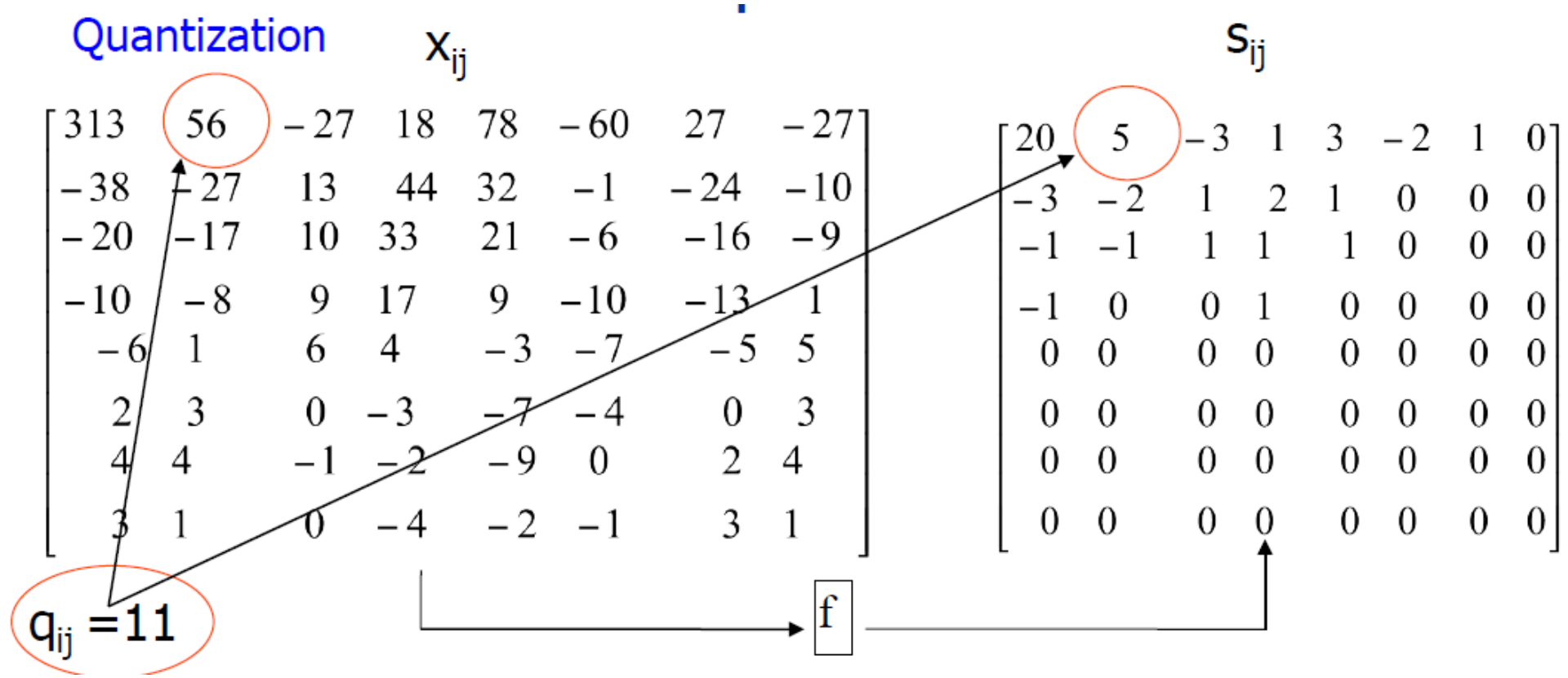
$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}$$

f

s_{ij}

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Compression



$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Compression

Quantization

x_{ij}

$$\begin{bmatrix} 313 & 56 & -27 & 18 & 78 & -60 & 27 & -27 \\ -38 & -27 & 13 & 44 & 32 & -1 & -24 & -10 \\ -20 & -17 & 10 & 33 & 21 & -6 & -16 & -9 \\ -10 & -8 & 9 & 17 & 9 & -10 & -13 & 1 \\ -6 & 1 & 6 & 4 & -3 & -7 & -5 & 5 \\ 2 & 3 & 0 & -3 & -7 & -4 & 0 & 3 \\ 4 & 4 & -1 & -2 & -9 & 0 & 2 & 4 \\ 3 & 1 & 0 & -4 & -2 & -1 & 3 & 1 \end{bmatrix}$$

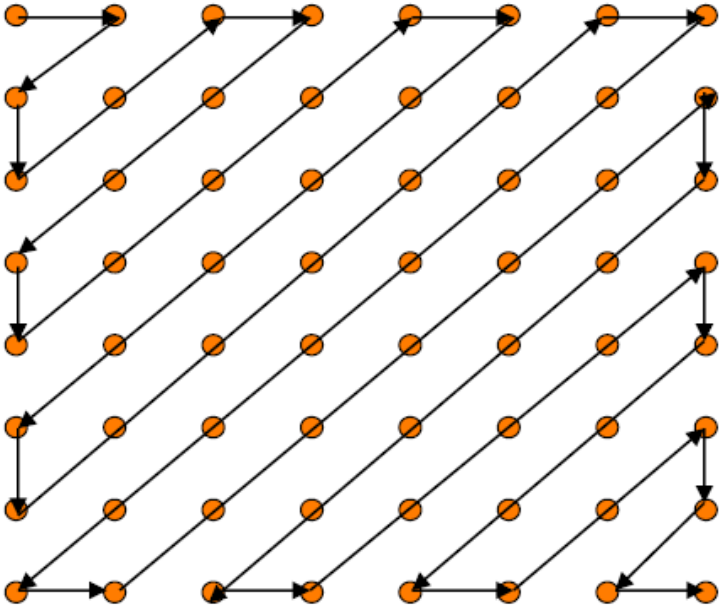
f

s_{ij}

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Compression

Quantization



Zigzag Scan

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

zigzag scan

(20,5,-3,-1,-2,-3,1,1,-1,-1,
0,0,1,2,3,-2,1,1,0,0,0,0,0,
0,1,1,0,1,EOB)

JPEG Compression

Differential Coding, Run Length Coding, Variable Length Coding

(20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

DC coefficient

AC coefficient

- DC coefficient : Differential coding
- AC coefficient : run-length coding (run, level)

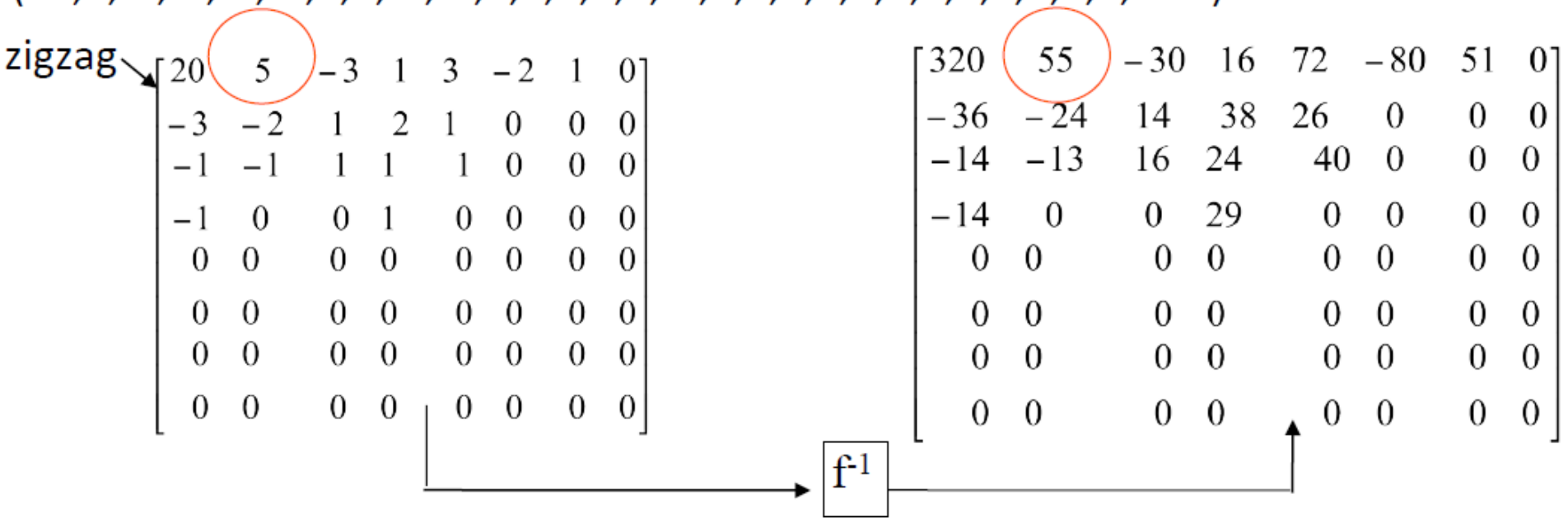
→ Huffman coding

JPEG Compression

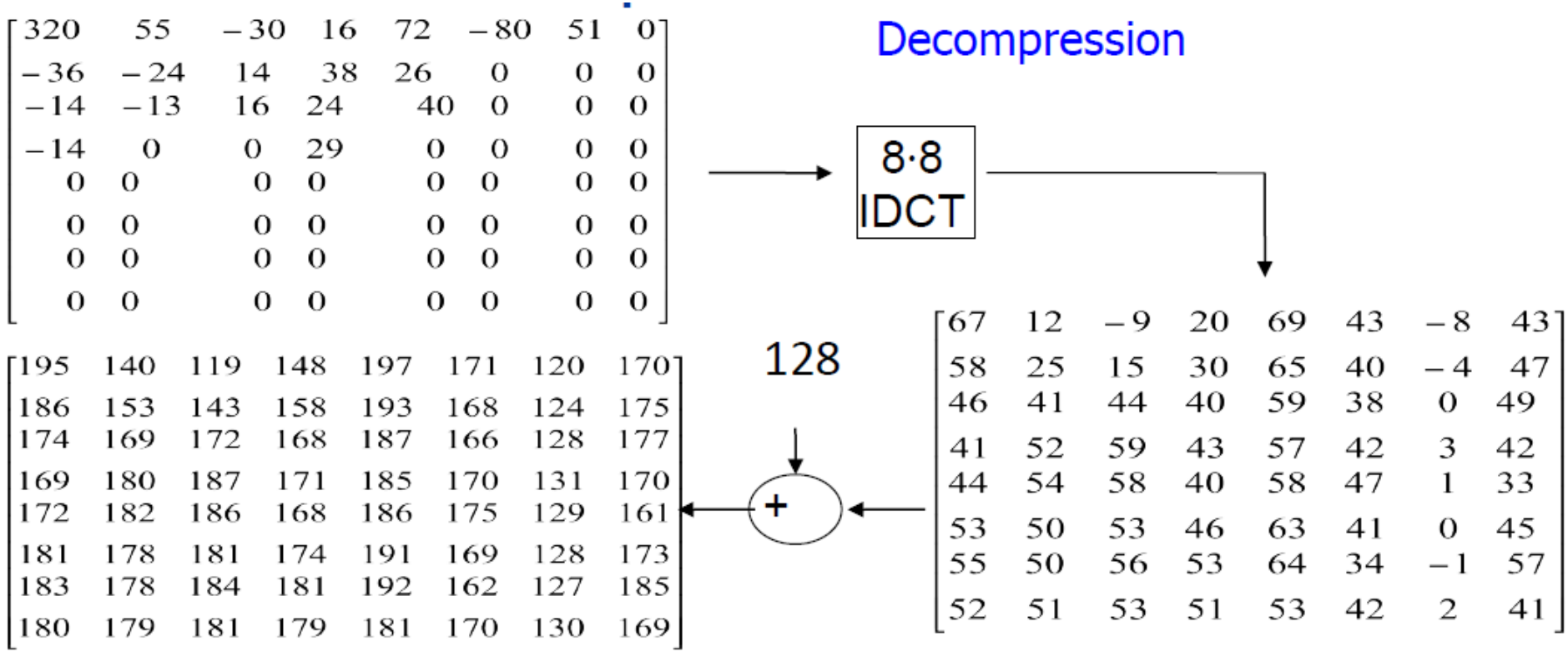
Decompression

(20,5,-3,-1,-2,-3,1,1,-1,-1,0,0,1,2,3,-2,1,1,0,0,0,0,0,0,1,1,0,1,EOB)

zigzag



JPEG Compression



$$\begin{bmatrix} 320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 \\ -36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\ -14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\ -14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 67 & 12 & -9 & 20 & 69 & 43 & -8 & 43 \\ 58 & 25 & 15 & 30 & 65 & 40 & -4 & 47 \\ 46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\ 41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\ 44 & 54 & 58 & 40 & 58 & 47 & 1 & 33 \\ 53 & 50 & 53 & 46 & 63 & 41 & 0 & 45 \\ 55 & 50 & 56 & 53 & 64 & 34 & -1 & 57 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

$$\begin{bmatrix} 195 & 140 & 119 & 148 & 197 & 171 & 120 & 170 \\ 186 & 153 & 143 & 158 & 193 & 168 & 124 & 175 \\ 174 & 169 & 172 & 168 & 187 & 166 & 128 & 177 \\ 169 & 180 & 187 & 171 & 185 & 170 & 131 & 170 \\ 172 & 182 & 186 & 168 & 186 & 175 & 129 & 161 \\ 181 & 178 & 181 & 174 & 191 & 169 & 128 & 173 \\ 183 & 178 & 184 & 181 & 192 & 162 & 127 & 185 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

JPEG Compression

$$X = \begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

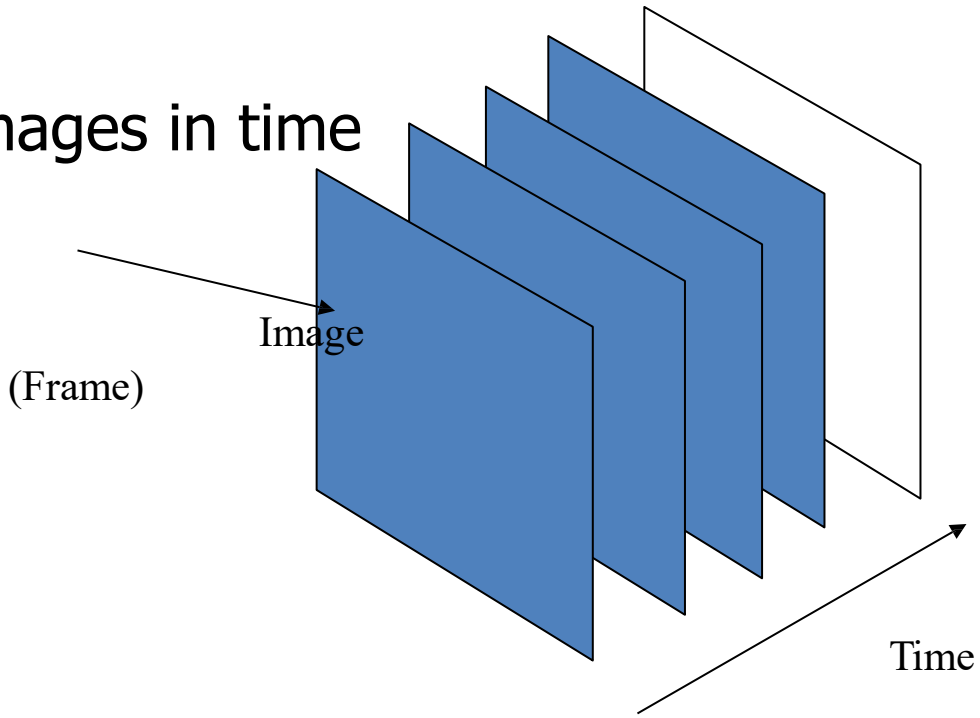
$$\hat{X} = \begin{bmatrix} 195 & 140 & 119 & 148 & 197 & 171 & 120 & 170 \\ 186 & 153 & 143 & 158 & 193 & 168 & 124 & 175 \\ 174 & 169 & 172 & 168 & 187 & 166 & 128 & 177 \\ 169 & 180 & 187 & 171 & 185 & 170 & 131 & 170 \\ 172 & 182 & 186 & 168 & 186 & 175 & 129 & 161 \\ 181 & 178 & 181 & 174 & 191 & 169 & 128 & 173 \\ 183 & 178 & 184 & 181 & 192 & 162 & 127 & 185 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

Distortion calculation: $MSE=||X-\hat{X}||^2$

Video Compression

Digital Video

Video is a sequence of images in time
- can be edited
- can be stored on any digital medium
- can be compressed



Video Compression

- The Need for Video Compression
- Huge data
- Example: High-Definition Television (HDTV)
- 1920x1080
- 30 frames per second (full motion)
- 8 bits for each three primary colors →Total 1.5 Gb/sec!
- Channel bandwidth 19.2 Mb/sec
- Reduced to 18 Mb/sec w/audio + control …

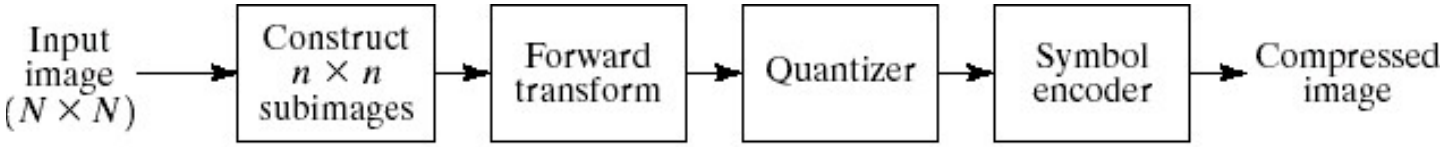
→Compression rate must be 83:1!

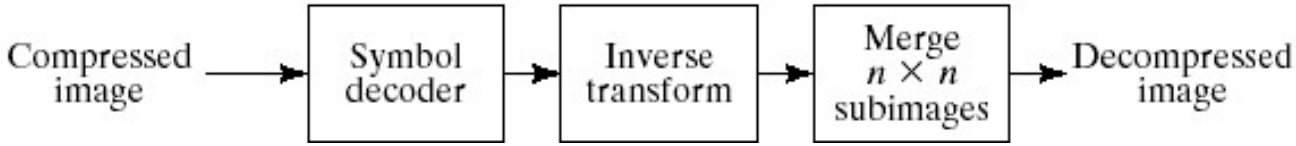
Video Compression

Image Compression: Transform Coding-> JPEG

Pipeline

Compression





Decompression

Video Compression

MJPEG (Motion JPEG)

- Each frame can be compressed as single image.
- Compression is achieved only due to the spatial redundancy in the frame.
- Takes care of intra-frame redundancy

Video Compression

Anything else that can be done?

- What about temporal redundancy or inter frame redundancy?

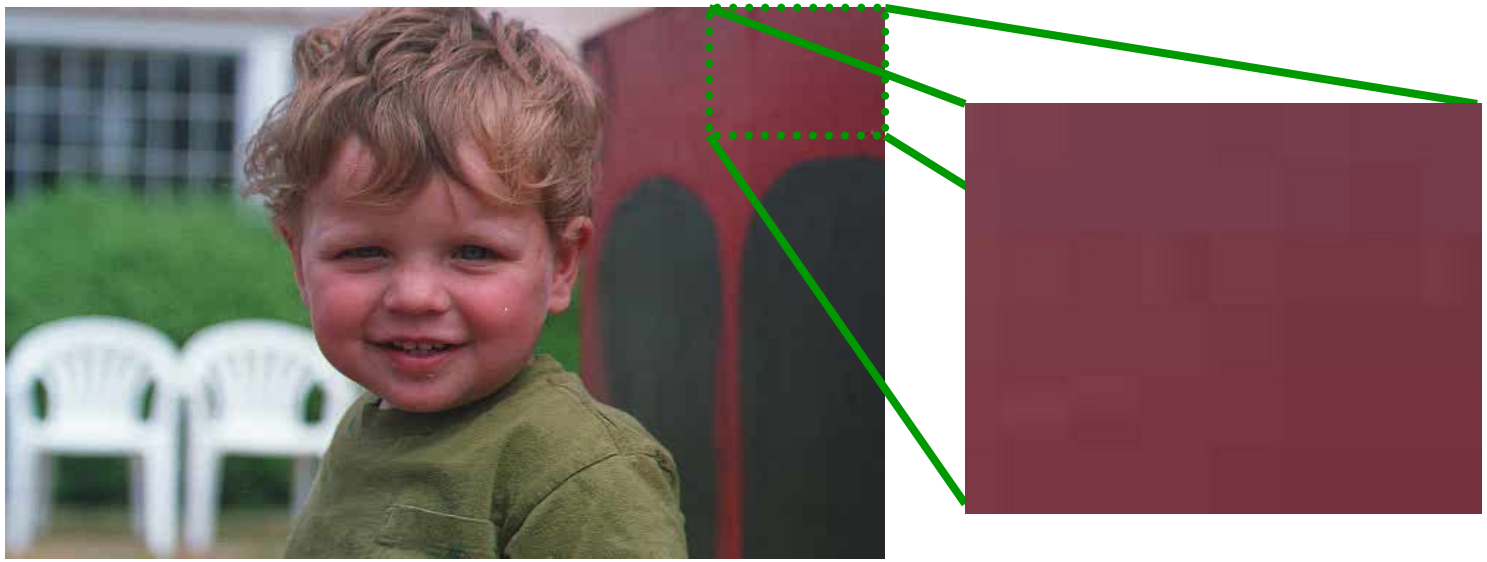
MPEG (Motion Picture Experts Group)

- What about irrelevancy – perceptually unimportant?

Video Compression

Spatial Redundancy

Take advantage of similarity among most neighboring pixels



Video Compression

Temporal Redundancy

Video: Sequence of images in time (that are related!)

Take advantage of similarity between successive frames

950

951

952

Video Compression

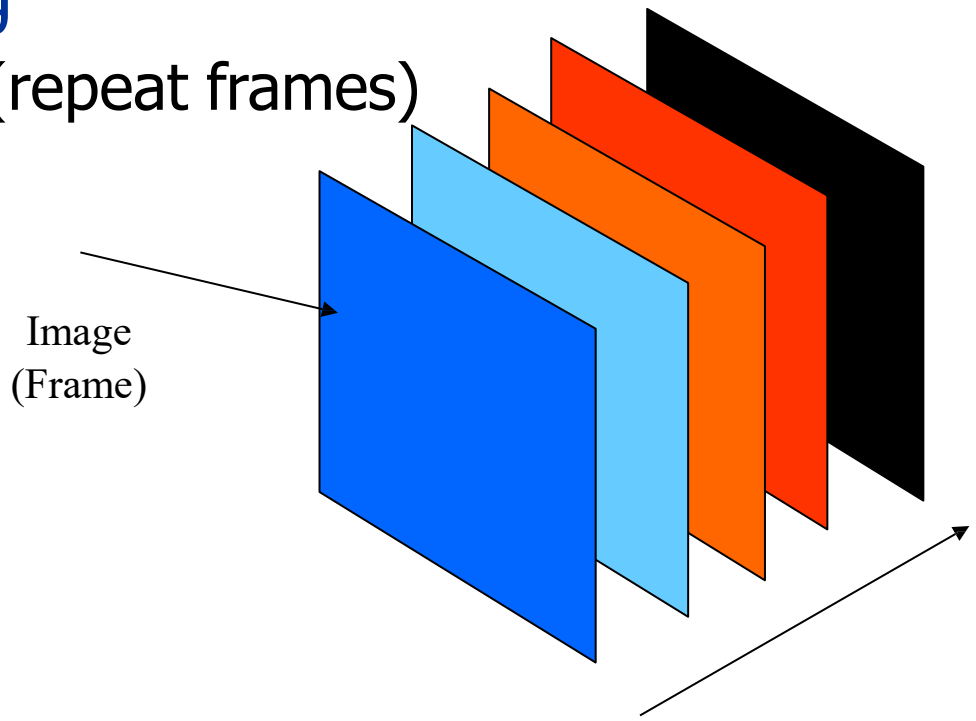
Intuitive Methods

- Subsampling
  - Drop frames
- Differencing
  - Differential coding of pixels
- Block Differencing
  - Differential coding of blocks (big pixels)
- Motion Compensation
  - Figure out the motion vector and compensate for it

Video Compression

Subsampling

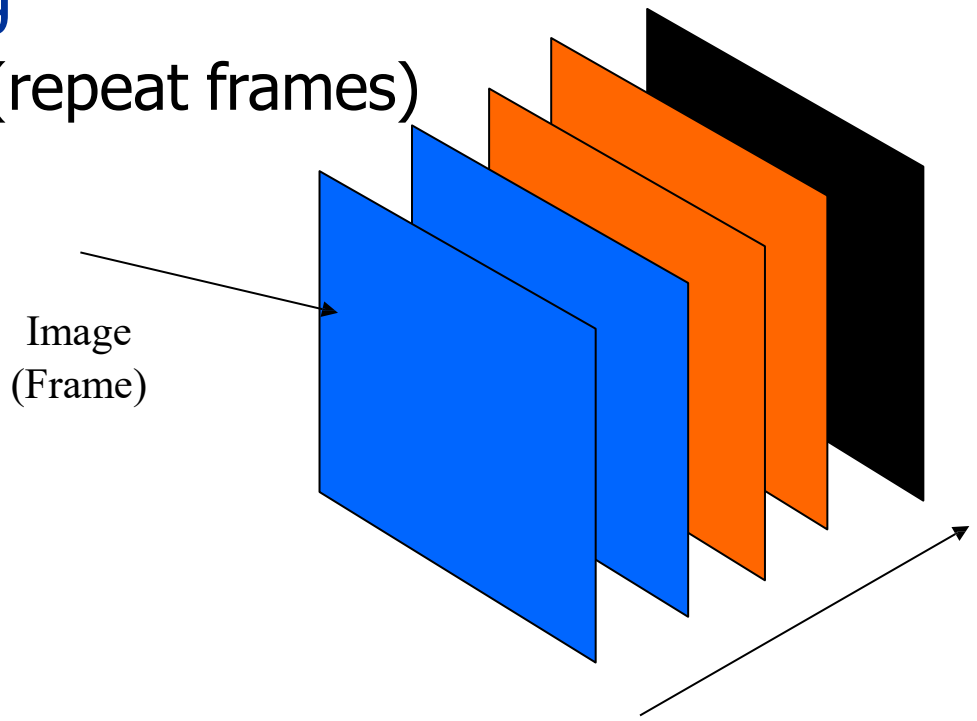
Drop frames (repeat frames)



Video Compression

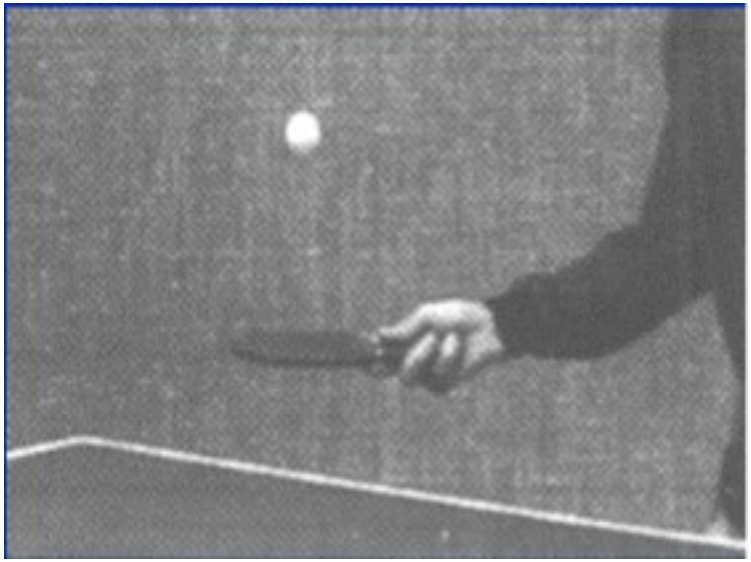
Subsampling

Drop frames (repeat frames)

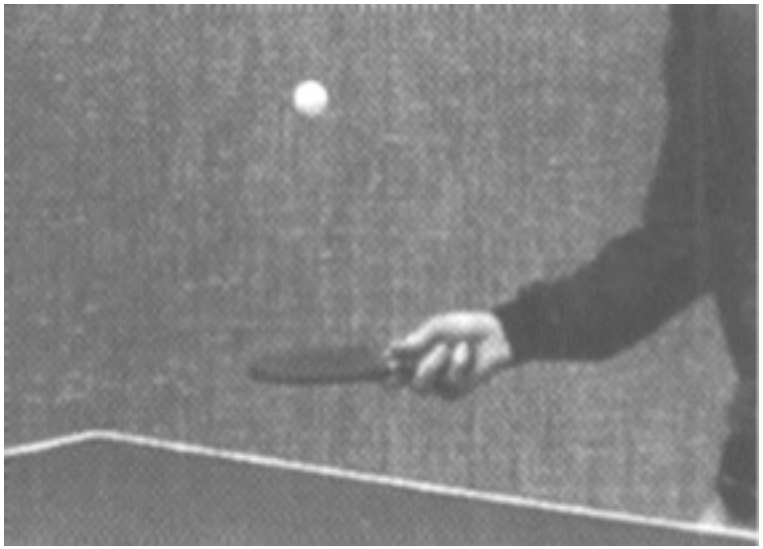


Video Compression

Differencing



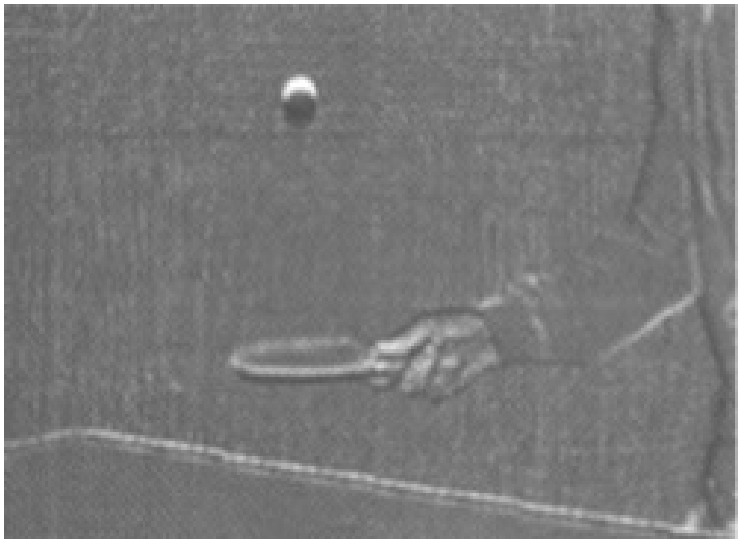
Frame N



Frame N+1

Video Compression

Differencing

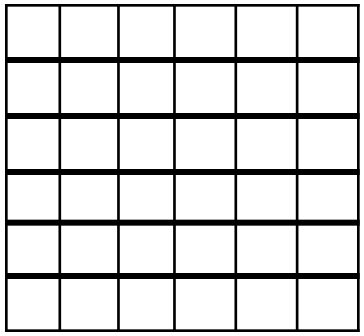


Difference frame

Video Compression

Block Differencing

- Frame is divided into non-overlapping blocks
- Block level comparison rather than pixel level to decide which blocks for the difference is to be coded
- May work when the motion is relatively small of foreground objects
- If the motion is large and not limited to portion of image then it may not be effective



Video Compression

Motion Compensation

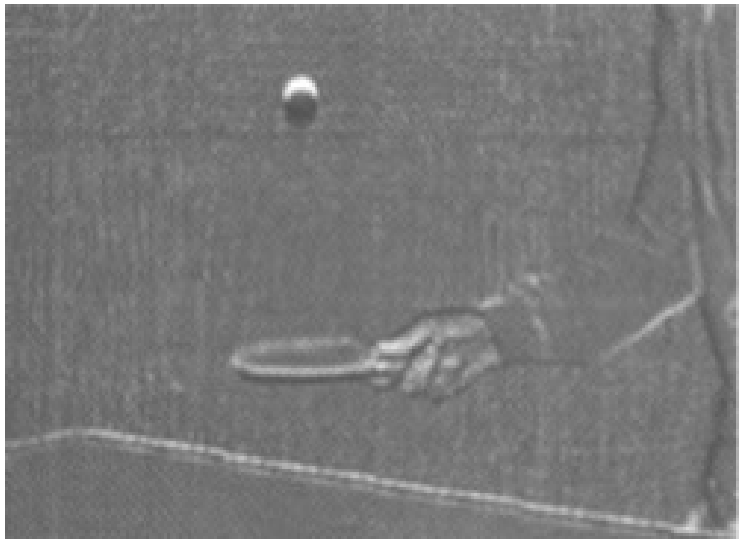
- Simple frame difference will fail if there is a significant motion
- Should account for the motion

Motion-compensated (MC) prediction

- How can we estimate motion?

Video Compression

Motion Compensation



Difference frame without motion prediction

Difference frame with motion prediction

Video Compression

Motion Estimation

•A possible approach

Segment video into moving objects Describe (model) object motion May be some what difficult

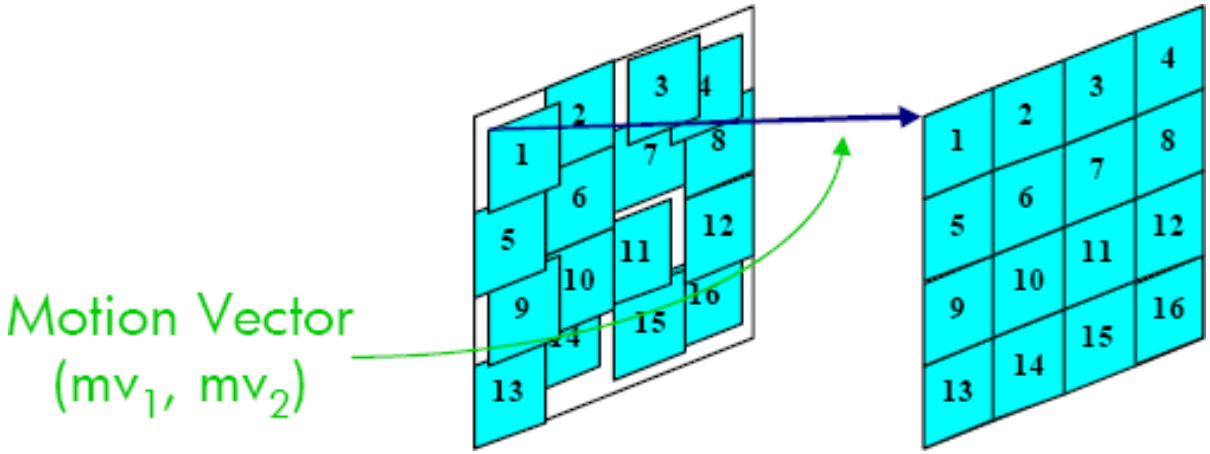
•Another (practical) approach

Block matching motion estimation

No object segmentation and identification required

Good performance

Motion Estimation



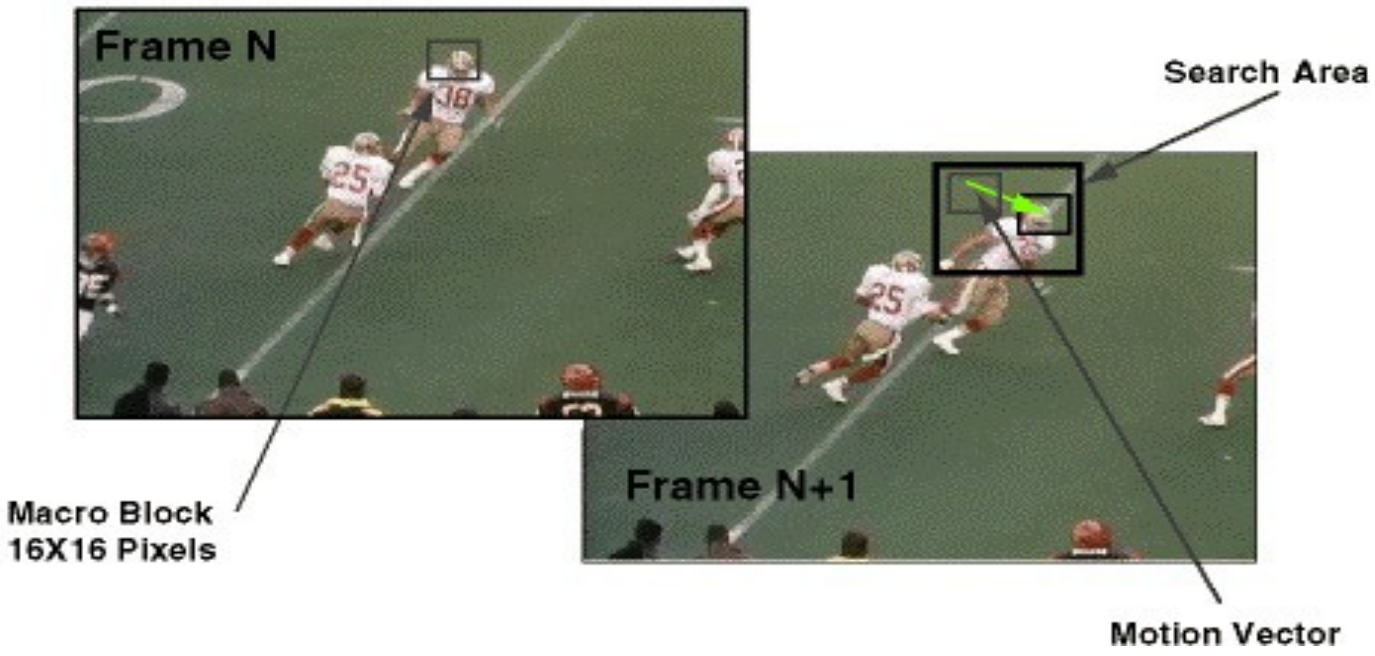
Video Compression

Motion Estimation

- Translation motion model
- All pixels within the block have the same motion
- Motion is estimated using only luminance
- The motion vector is encoded in place of the target block itself.
- Fewer bits are required to code a motion vector

Video Compression

Motion Compensation

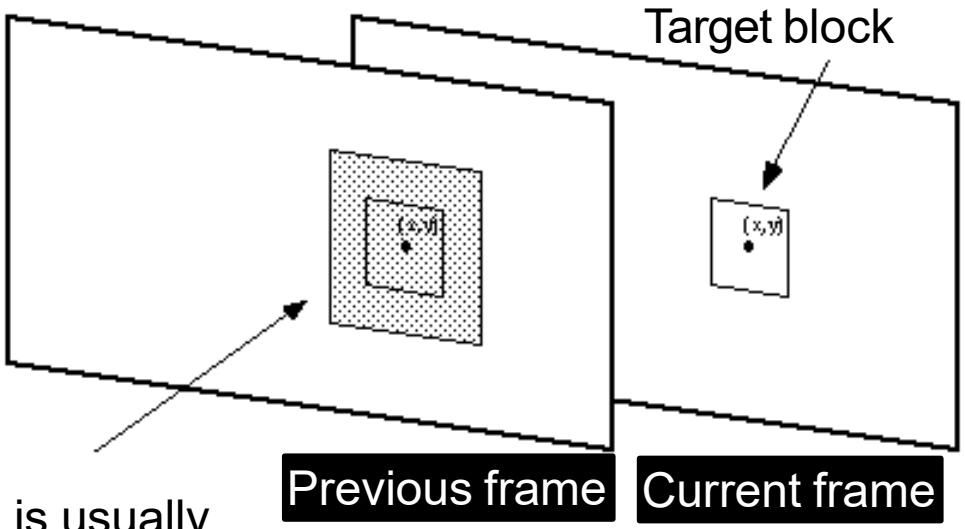


Video Compression

Motion Estimation

Issues: Block size ?
Search range ?

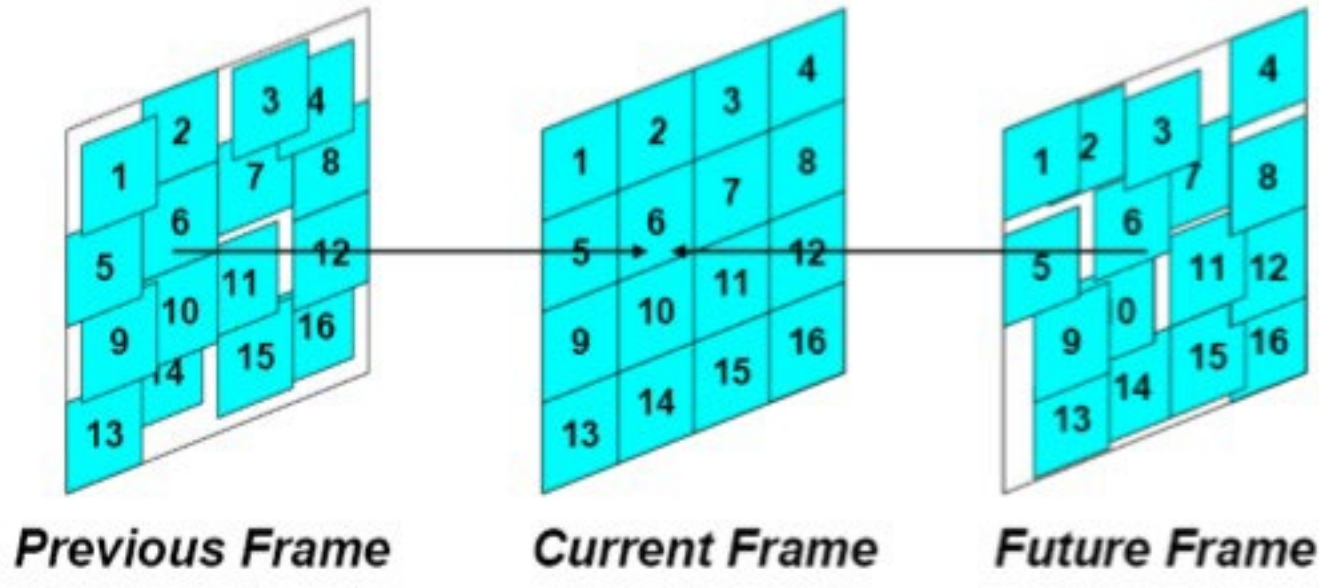
Motion vector accuracy ?
Complex motion ?



Search area in previous frame is usually limited to a region close to the target block

Video Compression

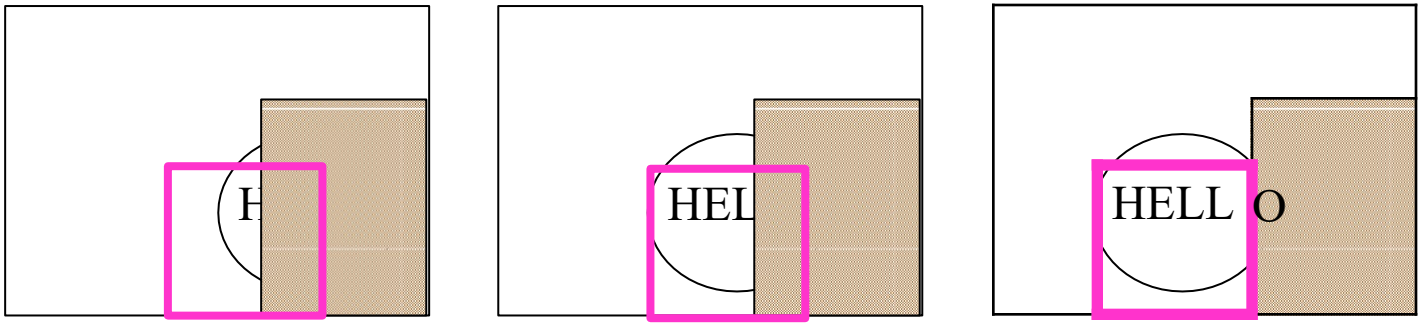
- Motion Estimation
- Forward, Backward, Bidirectional



Video Compression

Motion Estimation

Example



Frame N-1 Frame N Frame N+1

Video Compression

Motion Estimation

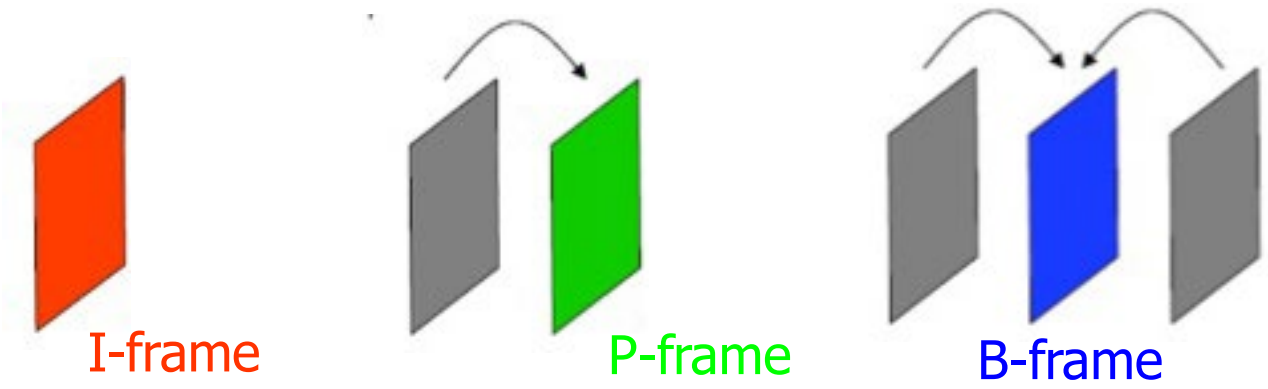
As a general scheme a block in current frame can be estimated from a block in

- Previous frame
- Future frame
- Average of a block from the previous frame and a block from the future frame
- Neither (no prediction)

Video Compression

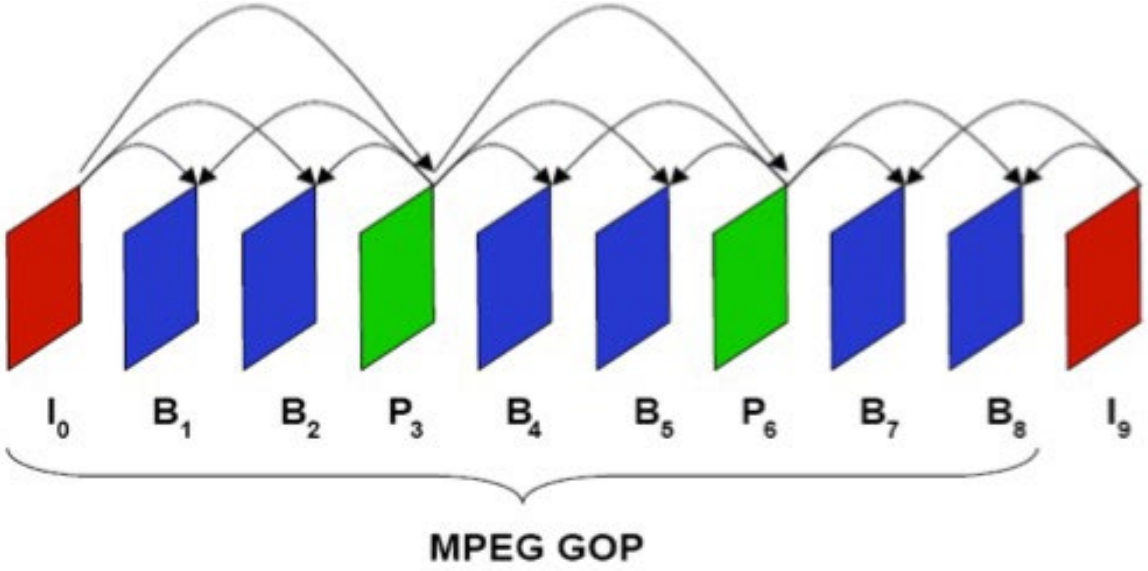
Three types of coded frames

- I-frame: Intra coded frame, coded independently
- P-frame: Predicitve coded frame, coded based on previously coded frame
- B-frame: Bi-directionally predicted frame, coded based on previous and future coded frames



Video Compression

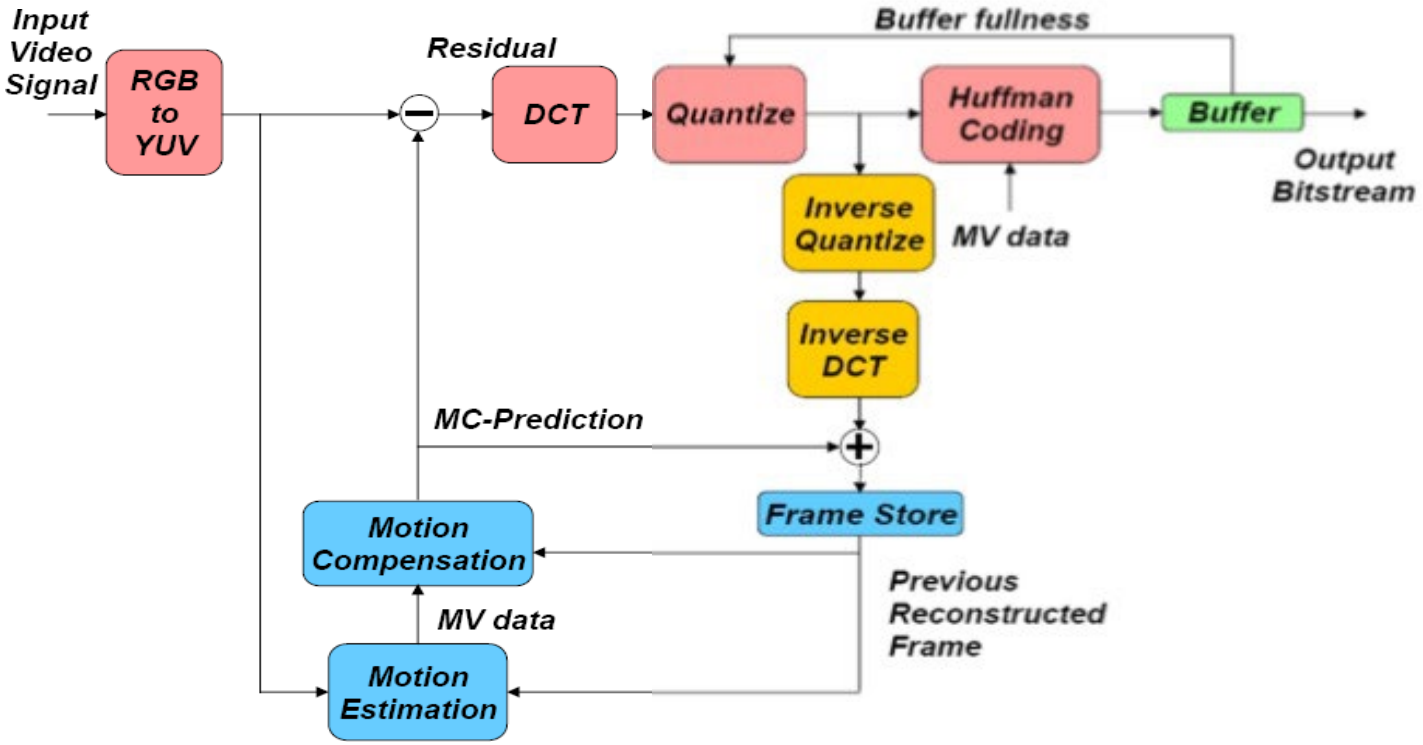
Three types of coded frames



Display order: I_0, B_1, B_2, P_3, B_4, B_5, P_6, B_7, B_8, I_9
Transmission Order: I_0, P_3, B_1, B_2, P_6, B_4, B_5, I_9, B_7, B_8

Video Compression

Encoding



Video Compression

Decoding