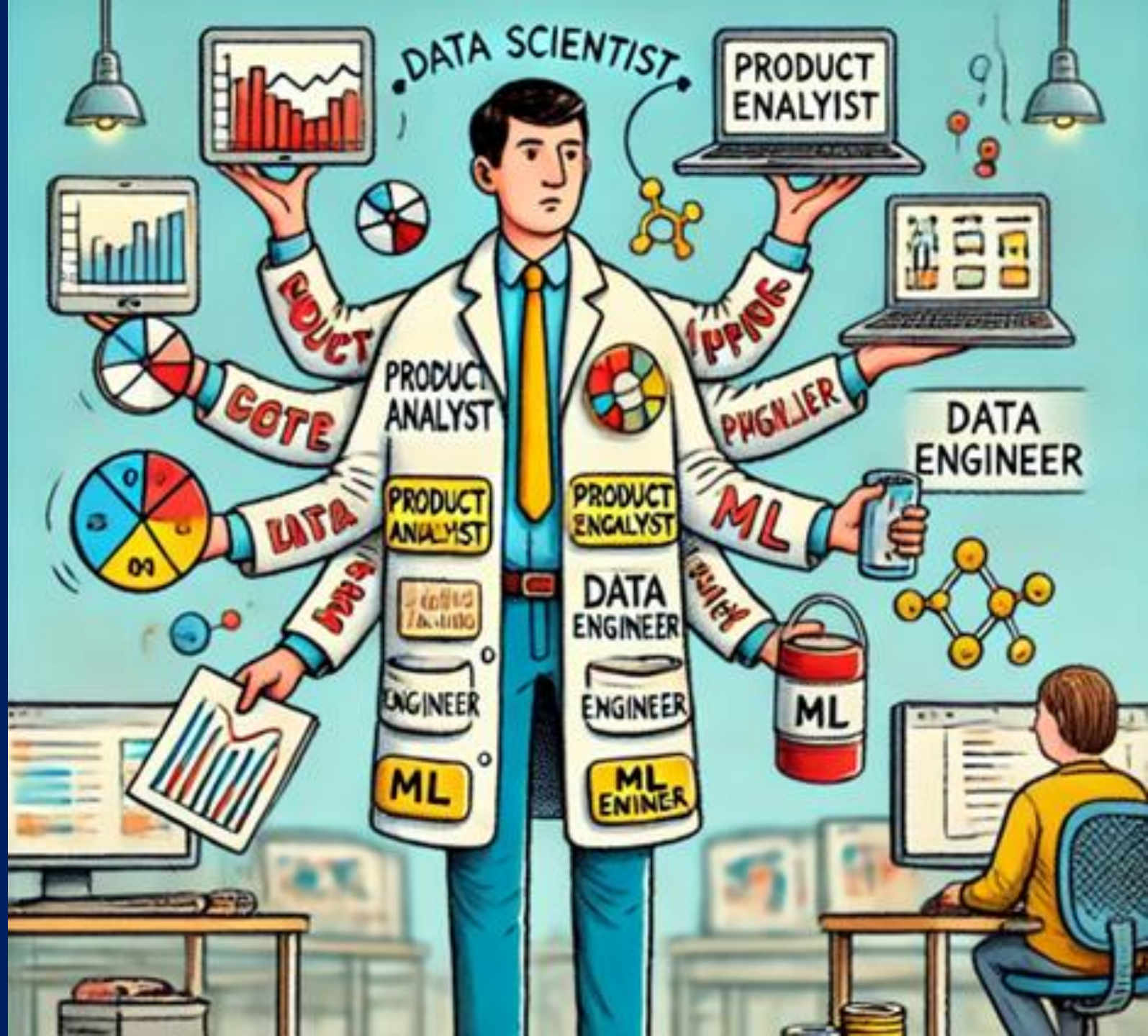
CS 51 Computer Science Principles

Module 3: Data, Internet, Computer and Programming

Unit 2: Data and Information

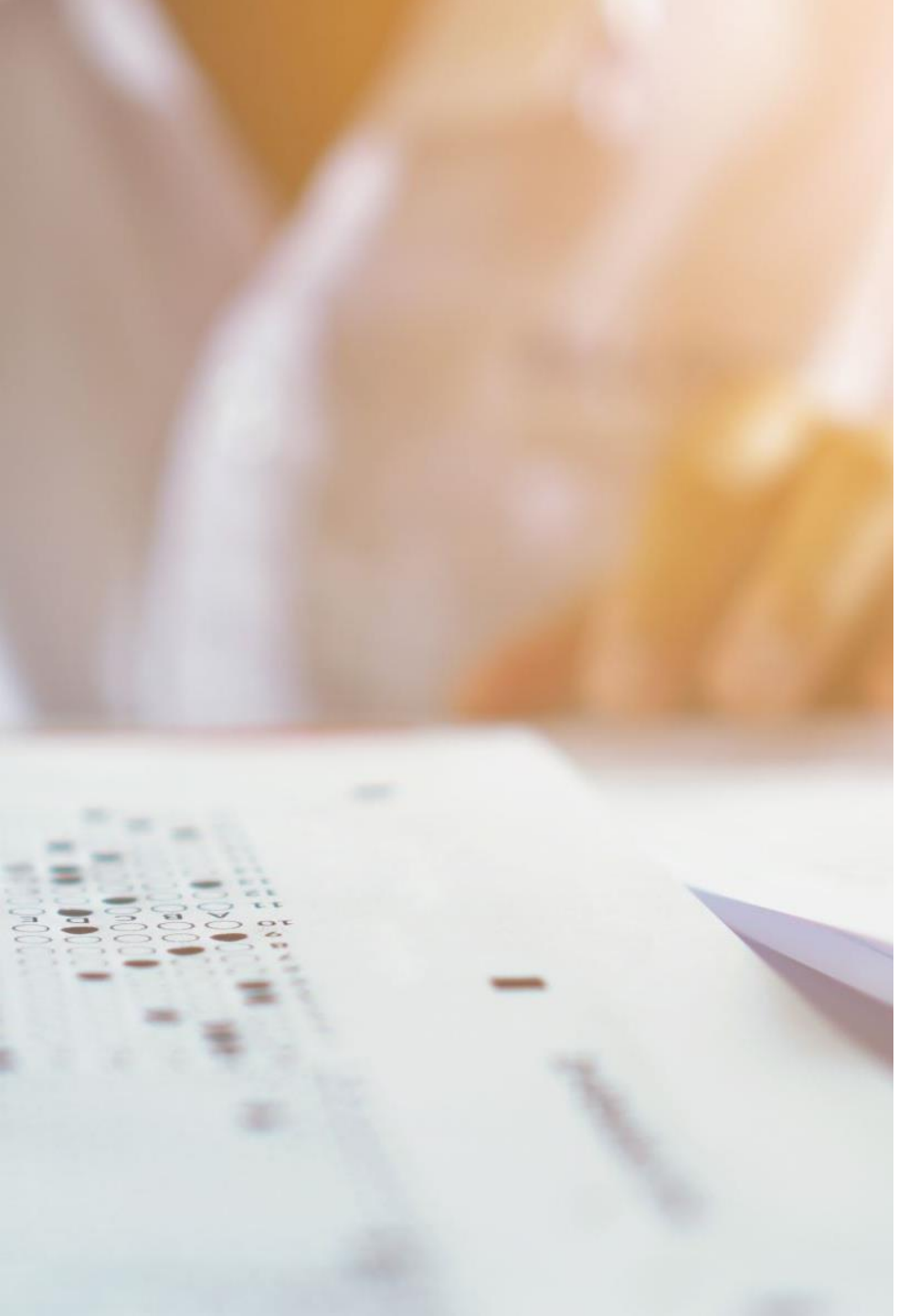
LECTURE 3 DATA COMPRESSION

DR. ERIC CHOU
IEEE SENIOR MEMBER



The One Thing You Need to Know About this Big Idea:

- This unit is all about how computers represent data, and how they can store and process ever-increasing quantities of it.



Unit Overview:

Exam Weighing:

- 17-22% of the AP Exam
- Practically, this translates to about 20 questions on the test.

Data Compression

LECTURE 1

Data Compression

- The need for hexadecimal representations of binary values brings up an important point about digital data: there's a lot of it, and it can quickly munch into the number of bytes you have for storage (no pun intended.) It's also notoriously difficult to send files like photos or videos over text or email because of how large they are.
- That's why data compression is needed. **Data compression** is the process (or processes) by which the size of shared or stored data is reduced. This in essence reduces the number of bits.

Data Compression

- The amount you can shrink your file by depends on two things:
  - the amount of redundancy, or repeated information, that you can remove in your original data
  - the method you use to compress your file
- Many data compression methods work by using symbols to shorten the data.

Data Compression Examples

- A simple form of data compression is known as run-length encoding. It works by replacing repeating data, such as colors in an image or letters in a document, with a run that represents the number and value of the repeated data. For example, the string "FFFFFIIIIIIVVVVVVVVEEEE" would be stored as 5F6I7V4E, greatly reducing the number of bytes needed to store it.
- Run-length encoding is used to compress some simple images, such as bitmaps. It is also used by fax machines.
- Another method of data compression that replaces repeating data with symbols is known as the LZW compression algorithm. It's used to compress text and images, most notably in GIFs.
- Let's take a look at two of the most common compression types: lossless and lossy.

Lossless vs. Lossy Data Compression:

- Lossless compression algorithms allow you to reduce your file size without sacrificing any of the original data in the process. You're generally able to restore your original file if you want. Run-length encoding and the LZW algorithm are both examples of lossless compression because they only shorten data to compress it, and all the information remains the same.
- If your main concern is the quality of your file or if you need to be able to reconstruct your original file, lossless algorithms are usually the better option.
- This type of storage might be important in databases, where a difference in a compressed versus an uncompressed file could skew the information being represented.

Lossless vs. Lossy Data Compression:

- This concern about skew also applies to both medical and satellite imaging, where small differences in the data could have large impacts. Many software downloads also use lossless compression methods because the programs need to be recreated exactly on your computer in order to work.
- In contrast, lossy compression algorithms sacrifice some data in order to achieve greater compression than you can achieve with a lossless method. They usually do this by removing details, such as replacing similar colors with the same one in a photo.

An image of tacos before and after lossy compression. Although the image on the right uses 62% less data than the image on the top, the two images look practically identical.

Lossless vs. Lossy Data Compression:

- If your main concern is minimizing how big your file is or how long it'll take to send or receive it, go with a lossy method! A lot of lossy compression methods make changes that are barely detectable or even undetectable to your average viewer, and they can save you a lot of space. It's commonly used in photo, audio, and visual compression, especially for downloading purposes.
- Although there are two main types of data compression, you don't have to choose just one. Indeed, many modern compression software systems use a combination of the two methods in some way.

Text Compression

LECTURE 2

Vocabulary

- **Heuristic** - a problem solving approach (algorithm) to find a satisfactory solution where finding an optimal or exact solution is impractical or impossible.
- **Lossless Compression** - a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.
- **lossy compression:** - discards some data to make an imperfect reproduction – as in jpeg formats

Vocabulary

- **Compress:** - to decrease the number of bits used to represent a piece of information

Lossless Compression

- The basic principle behind compression is to develop a method or protocol for using fewer bits to represent the original information.
- The way we represent compressed data in this lesson, with a "dictionary" of repeated patterns is similar to the **LZW compression** scheme, but it should be noted that LZW is slightly different from what you will do in this lesson.
- You will invent your own way here. LZW is used not only for text (zip files), but also with the GIF image file format.

Heuristics

- The lesson touches on computationally hard problems and heuristics but please note that computationally hard problems and heuristics will be revisited later on. A general understanding is all that's needed from this lesson.
- There is no single correct way to compress text using the method we use in this lesson because
  - **a) there is no known algorithm for finding an optimal solution, and**
  - **b) we don’t even know a way to verify whether a given solution is optimal.**
- There is no way to prove it or derive it beyond trying all possibilities by brute force. This is an example of an algorithm that cannot run in a “reasonable amount of time” - one of the CSP learning objectives.

Video

- Video: Text Compression with Aloe Blacc – Video

- Code Studio: Human Text Compression
- Note: replace, replaceAll functions in programming languages.

Activity: Human Data Compression

- Decoding the message in the warm-up activity is very similar to tracing a sequence of function calls in a **program**.
    1. When you send text messages to a friend, do you spell every word correctly?
    2. Do you use **abbreviations** for common words? List as many as you can.
    3. Write some examples of things you might see in a text message that are not proper English.
    4. Why do you use these abbreviations? What is the benefit?“

Purpose of Using Human Compressed Text

1. to save characters/keystrokes
2. to hide from parents/teachers
3. to be cool, clever, funny
4. to “speak in code”
5. to say the same thing in less space

Human Data Compression

- Today's class is about compression
  When you abbreviate or use coded language to shorten the original text, you are “compressing text.”
- Computers do this too, in order to save time and space.
- The art and science of compression is about figuring out how to represent the SAME DATA with FEWER BITS.

Why is this important?

- One reason is that **storage space** is limited and you'd always prefer to use fewer bits if you could. A much more compelling reason is that there is an upper limit to how fast bits can be transmitted over the Internet.
- What if we need to send a large amount of text faster over the Internet, but we've reached the physical limit of how fast we can send bits?
- Our only choice is to somehow capture the same information with fewer bits; we call this compression.

Data Compression

- **Data compression** is the process of modifying, encoding or converting the bits structure of data in such a way that it consumes less space on disk.
- It enables reducing the storage size of one or more data instances or elements. Data compression is also known as **source coding** or **bit-rate reduction**.

Activity Guide: Decoding the Message

- Decode this Mystery Text
- Make partners or work individual
- **Task:** What was the original text?

How Much?

40% compressed!

Original:	93 characters
Compressed:	56 characters
Difference:	**37 characters (~40%)**

Original Message

93 characters

Pitter_patter_pitter_patter_listen_to_the_rain_pitter_patter_pitter_patter_on_the_window_pane

Compressed

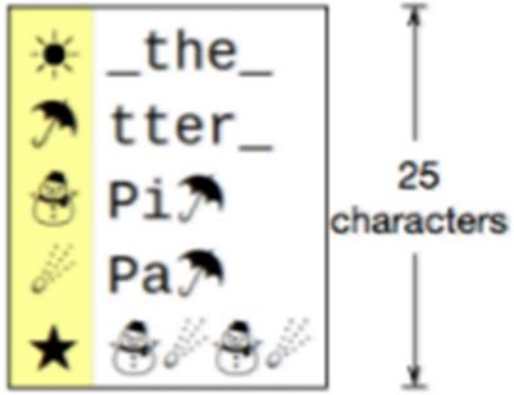
56 characters

★listen_to☀rain_★on☀window_pane

31 characters

Total number of characters needed to represent compressed version is:

31 (message) + 25 (key) = 56



Important Note

- The compressed poem is not just this part: ⭐listen_to 🌞rain_⭐on_🌞window_pane if you were to send this to someone over the internet, they would not be able to decode it.
- The full compressed text includes BOTH the compressed text and the key to solve it.
- Thus, you must account for the total number of characters in the message plus the total number of characters in the key to see how much you've compressed it over the original.

Activity

- Now you're going to get to try your hand at compressing some things on your own.
- Use the Text Compression Widget

Pair Activity

Each pair will compress a poem:

So wake me up . . .
A tutor who tooted the flute . . .
She sells sea shells . . .
I know an old lady . . .
Pease porridge hot . . .
I need a dollar . . .
The Man . . .

One answer for A Tutor

A_🪽 *who*🌞ed‹flute_Tried☆★Said‹🤶☂️their_🪽,"*I*sit_harder★Or☆☂️🌞?"

The dictionary:
🌞 toot
☂️ to
🤶 two
🪽 tu☂️r
★☂️🌞_
☆☂️🪽🤶🌞ers
‹ *the*

Compression Numbers:
Compressed text size: 66 bytes
Dictionary size: 41 bytes
Total: 107 bytes
Original text size: 148 bytes
Compression: 27.7%

Process:
1) Looked for parts of word that repeated (i.e. "th" in "the", "that", "those", etc)
2) Looked for phrases that repeated (used symbols already in dictionary, if possible)
3) Looked for words that repeated (using symbols in dictionary, if possible)

Challenge:

Challenge: compress your assigned poem as much as possible.

- Compare with other groups to see if you can do better.
- Try to develop a general strategy that will lead to a good compression.

What makes doing this compression hard?

- You can start in lots of different ways. Early choices affect later ones. Once you find one set of patterns, others emerge.
- **There is a tipping point:** you might be making progress compressing, but at some point the scale tips and the dictionary starts to get so big that you lose the benefit of having it. But then you might start re-thinking the dictionary to tweak some bits out.

What can be the compression scheme?

- Do we think that these compression amounts that we've found are the best? Is there a way to know what the best compression is?
- We probably don't know what's best.
- There are so many possibilities it's hard to know. It turns out the only way to guarantee perfect compression is brute force. This means trying every possible set of substitutions. Even for small texts this will take far too long. The "best" is really just the best we've found so far. [Are there ways to verify optimality?]

What can be the compression scheme?

- But is there a process a person can follow to find the best (or a pretty good) compression for a piece of text?
- Yes, but it's imprecise – let's see what happens next!

Heuristic versus Exact Solution

- In computer science there is a word for strategies to use when you're not sure what the exact or best solution to a problem is.
- Vocabulary: heuristic - a problem solving approach (typically an algorithm) to find a satisfactory solution where finding an optimal or exact solution is impractical or impossible.

Heuristic versus Exact Solution

- Do you think it's possible to describe (or write) a specific set of instructions that a person could follow that would always result in better text compression than your heuristic? Why or why not? Some compression programs (like zip) do a great job if the file is sufficiently large and has reasonable amounts of repetition.
- However, it is also possible to create a "compressed file" that is larger than the original because the heuristic does work in every single case.

Heuristic versus Exact Solution

- Is there a way to know that a compressed piece of text is compressed the most possible?
- There is no perfect solution.
- The size and shape of the data will determine what the "best" answer is and we often cannot even be sure it is the best answer.
- **The efficiency of a compression algorithm is actually content-dependent**

Abstraction

- What do all the compressions have in common?
- **Pattern Recognition**
  and
- **Abstraction** (patterns referring to other patterns)

Discussion

- Will following this process always lead to the same compression? (i.e. two people following the process for the same poem, will result in the same compression?)
- No. It's imprecise, but still OK. The text still gets compressed, no matter what.
- Since there is no way to know what's best, all we need is a process that comes up with some solution, and a way to make progress.

Discussion

- Why do you want to compress anything? What's the point?
- It is useful for sending things faster or for smaller storage. It allows for optimization of limited resources.

Compression in the Real World (.zip)

- There is a compression algorithm called LZW compression upon which the common “zip” utility is based.
- Zip compression does something very similar to what you did today with the text compression widget.

Video

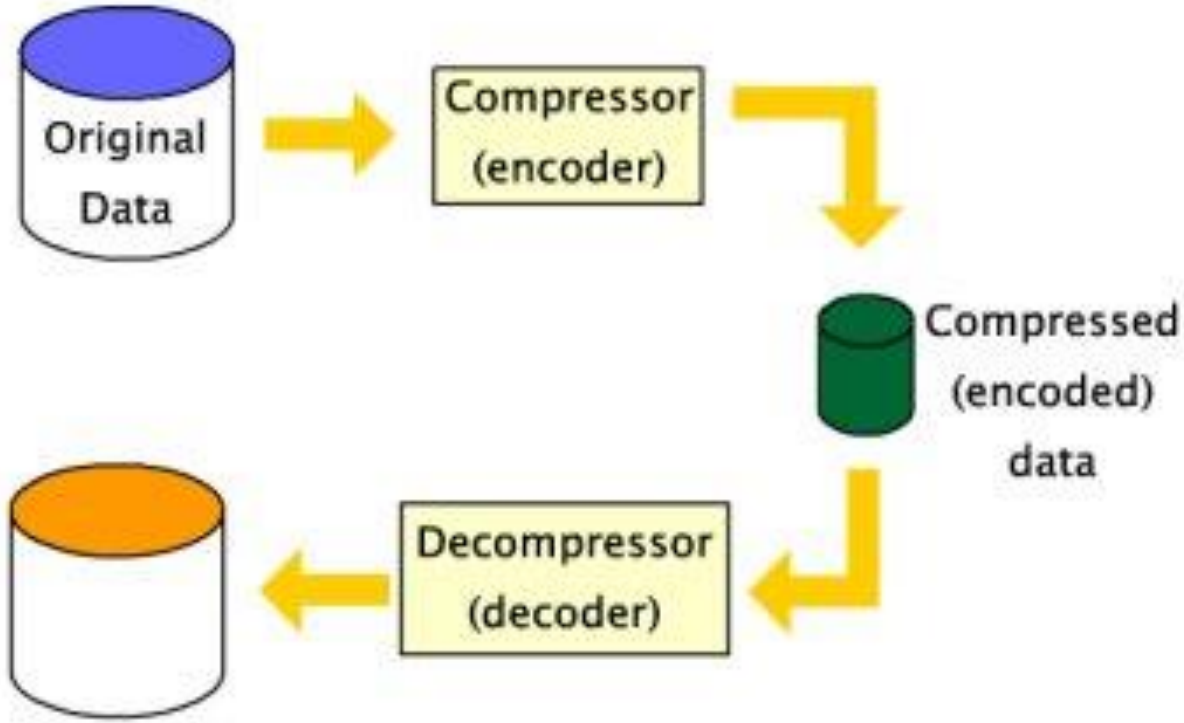
- Here is an video on Lempel–Ziv–Welch Algorithm
- You can see the algorithm doesn't compress it the most, but it is following a heuristic that will lead to better and better compression over time.

Send to Archive (Zip) and PeaZip

- Do you want to use zip compression for real?
- Most computers have it built in:
  Windows: select a file or group of files, right-click, and choose “Send To...Compressed (zipped) Folder.”

Lossless Compression

- The loop-back result is identical with the original.

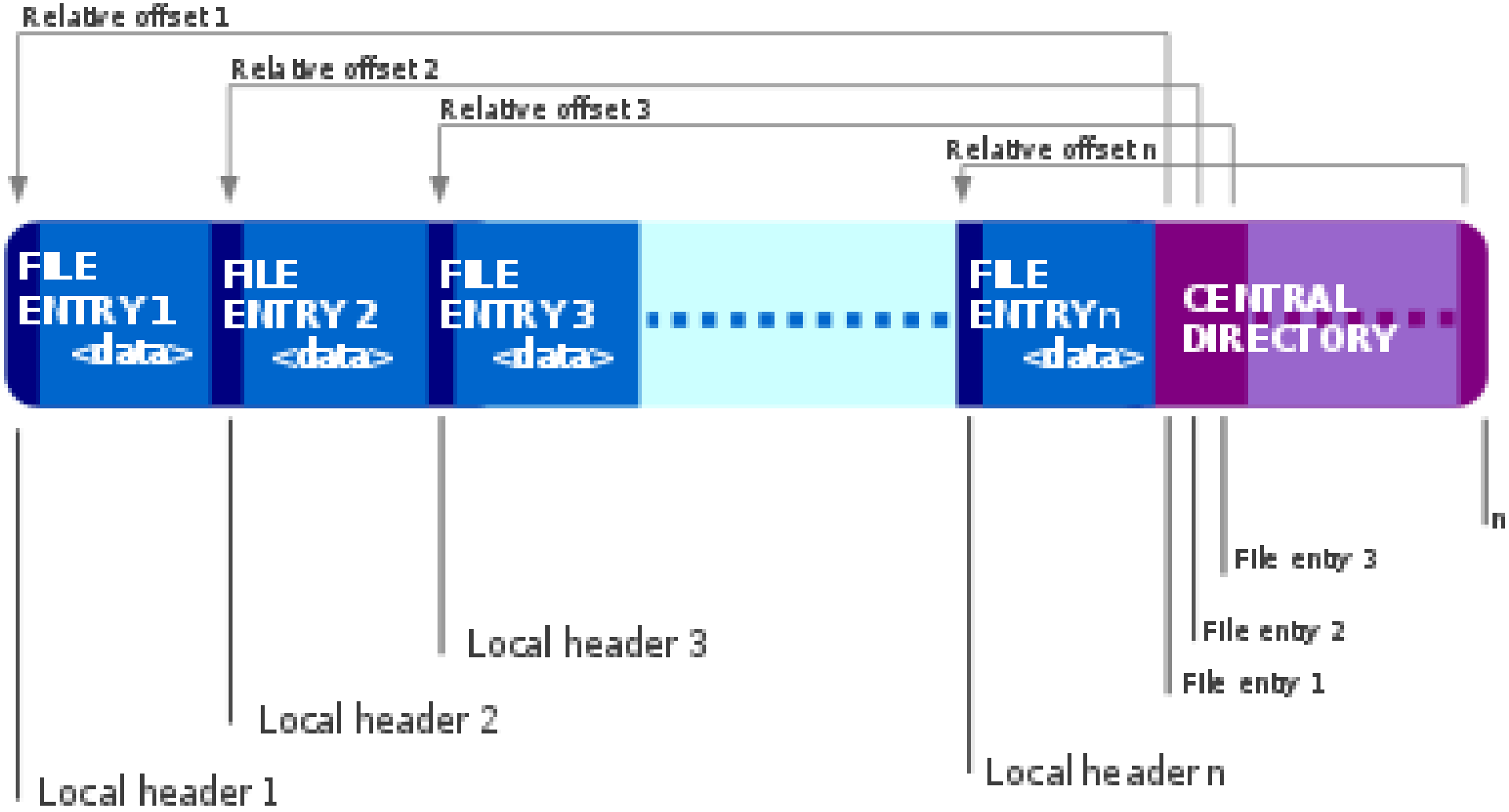


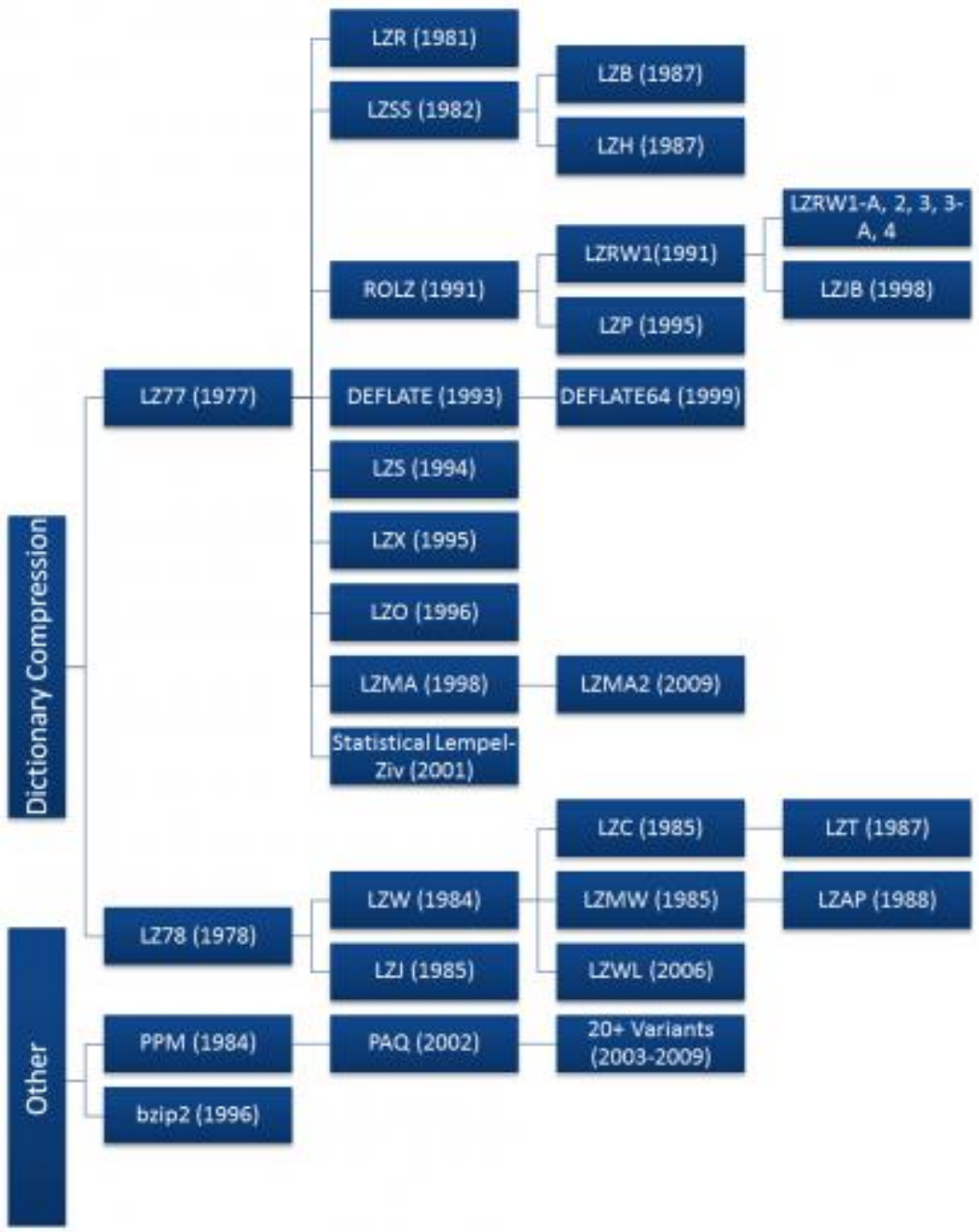
Zip

- Zip works really well for text, but only on large files. If you try to compress the simple hello.txt file we used in a previous lesson, you'll see the resulting file is actually bigger.
- Zip is meant for text. It might not work well on non-text files very well because they are already compressed or don’t have the same kinds of embedded patterns that text documents do.

Zip

- Zip works really well for text, but only on large files. If you try to compress the simple hello.txt file we used in a previous lesson, you'll see the resulting file is actually bigger.
- Zip is meant for text. It might not work well on non-text files very well because they are already compressed or don’t have the same kinds of embedded patterns that text documents do.


Relative offset 1
Relative offset 2
Relative offset 3
Relative offset n
FILE ENTRY 1 <data>
FILE ENTRY 2 <data>
FILE ENTRY 3 <data>
FILE ENTRY n <data>
CENTRAL DIRECTORY
Local header 1
Local header 2
Local header 3
Local header n
File entry 1
File entry 2
File entry 3
n

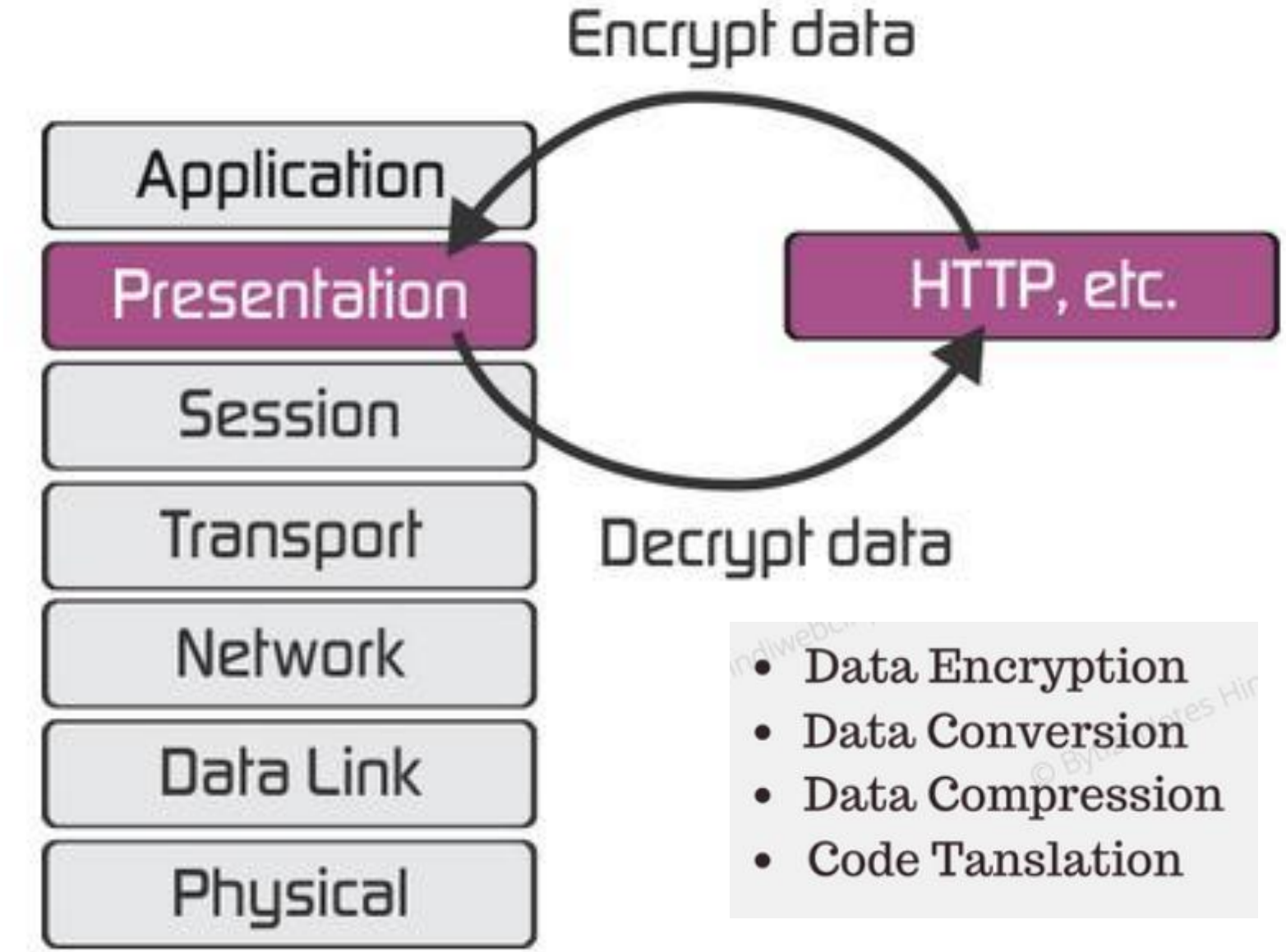


Assignment Activity Guide:

AG: Text Compression Heuristics
What does the fox say?
https://youtu.be/CuZBfX2mW7s

Dog goes "woof"
Cat goes "meow"
Bird goes "tweet"

And mouse goes "squeek"
Cow goes "moo"
Frog goes "croak"
And the elephant goes "toot"
Ducks say "quack"
And fish go "blub"
And the seal goes "ow ow ow"
But there's one sound
That no one knows
What does the fox say?
"Ring-ding-ding-ding-dingeringeding!
Gering-ding-ding-ding-dingeringeding!
Gering-ding-ding-ding-dingeringeding!"
What the fox say?
"Wa-pa-pa-pa-pa-pa-pow!
Wa-pa-pa-pa-pa-pa-pow!
Wa-pa-pa-pa-pa-pa-pow!"



- Data Encryption
- Data Conversion
- Data Compression
- Code Tanslation

	OSI Layer	TCP/IP	Datagrams are called
Software	**Layer 7** Application	HTTP, SMTP, IMAP, SNMP, POP3, FTP	Upper Layer Data
	Layer 6 Presentation	ASCII Characters, MPEG, SSL, TSL, Compression (Encryption & Decryption)	
	Layer 5 Session	NetBIOS, SAP, Handshaking connection	
	Layer 4 Transport	TCP, UDP	**Segment**
	Layer 3 Network	IPv4, IPv6, ICMP, IPSec, MPLS, ARP	**Packet**
Hardware	**Layer 2** Data Link	Ethernet, 802.1x, PPP, ATM, Fiber Channel, MPLS, FDDI, MAC Addresses	**Frame**
	Layer 1 Physical	Cables, Connectors, Hubs (DLS, RS232, 10BaseT, 100BaseTX, ISDN, T1)	**Bits**

Image Compression

LECTURE 3

Objectives

- Explain the difference between lossy and lossless compression.
- Identify common computer file types and whether they are compressed or not, and whether compression is lossy or lossless.
- Read a technical article on the web and sift its contents for targeted information.

Vocabulary

- **Lossless Compression** - a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.
- **Lossy Compression** - (or irreversible compression) a data compression method that uses inexact approximations, discarding some data to represent the content. Most commonly seen in image formats like .jpg.

File Format

- We've been looking at image file formats. And we've also seen text compression. Both of those attempted to render perfectly every piece of information.
- Both the image file format and the text compression scheme we used were lossless. Lossy compression schemes usually take advantage of the fact that a human is supposed to interpret the data at the other end, and human brains are good at filling the gaps when information is missing.

Video

- The Text Compression - Video (lossy in second half) video from a previous lesson also discusses lossy v. lossless

Note:

NOTE: The file extension you often see on a file (for example: myPhoto.jpg) is really just an indicator to the computer of how the underlying bits are organized, so the computer can interpret them. If you change the name of the file to myPhoto.gif, that does not magically change the underlying bits; all you've done is confuse the computer. It won't be able to open the file because it will attempt to interpret the file as a GIF when really the bits are in JPG format.

File Format

- All of these are specialized file formats in which some person or group decided how to organize (and in some cases, compress) the bits that make up the file type.
- There is nothing magical about them.

Code Studio

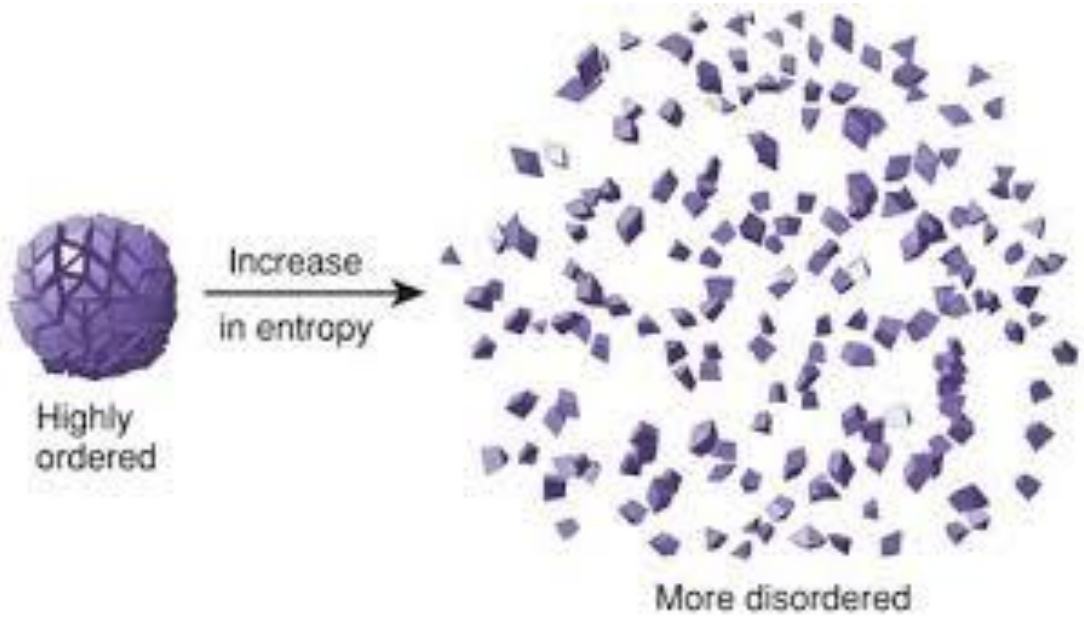
- Do the matching in Code Studio using your worksheet.
- Do the assessments/reflections.
- There is some homework to write about today.

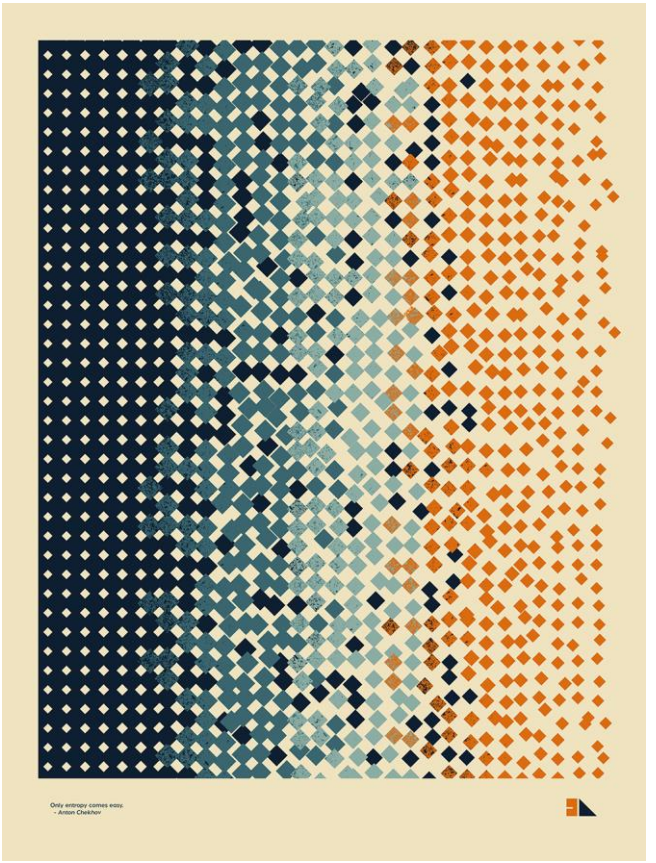
Huffman Coding

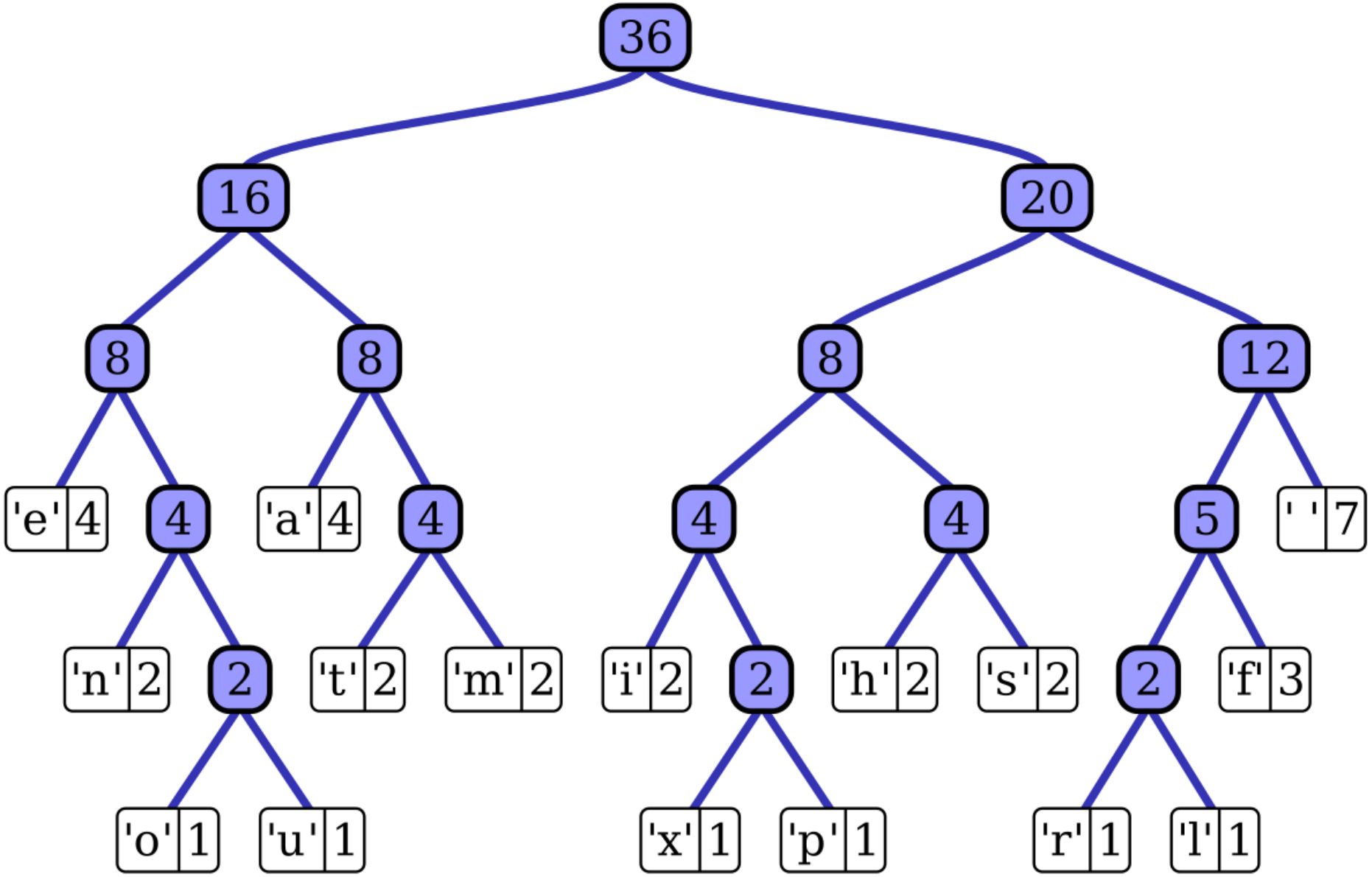
SECTION 5-1

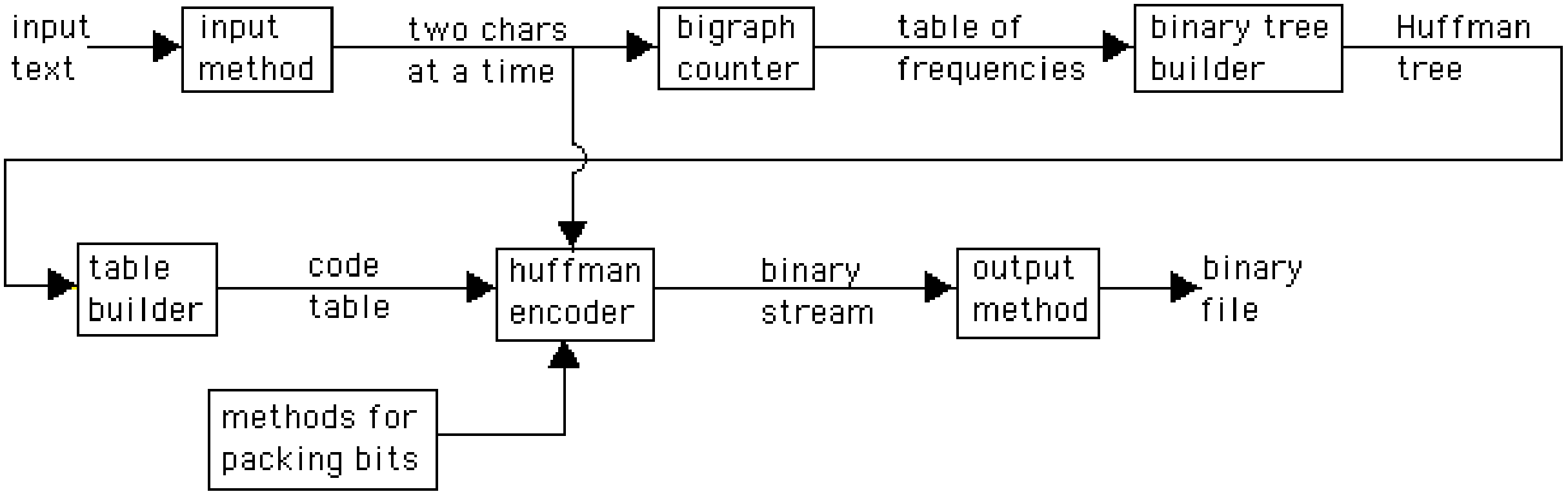
Huffman Encoding

- Good heuristic
- Entropy Encoding (Minimum Entropy)






36
16
20
8
8
8
12
'e' 4
4
'a' 4
4
4
4
5
' ' 7
'n' 2
2
't' 2
'm' 2
'i' 2
2
'h' 2
's' 2
2
'f' 3
'o' 1
'u' 1
'x' 1
'p' 1
'r' 1
'l' 1


input text
input method
two chars at a time
bigraph counter
table of frequencies
binary tree builder
Huffman tree
table builder
code table
huffman encoder
binary stream
output method
binary file
methods for packing bits

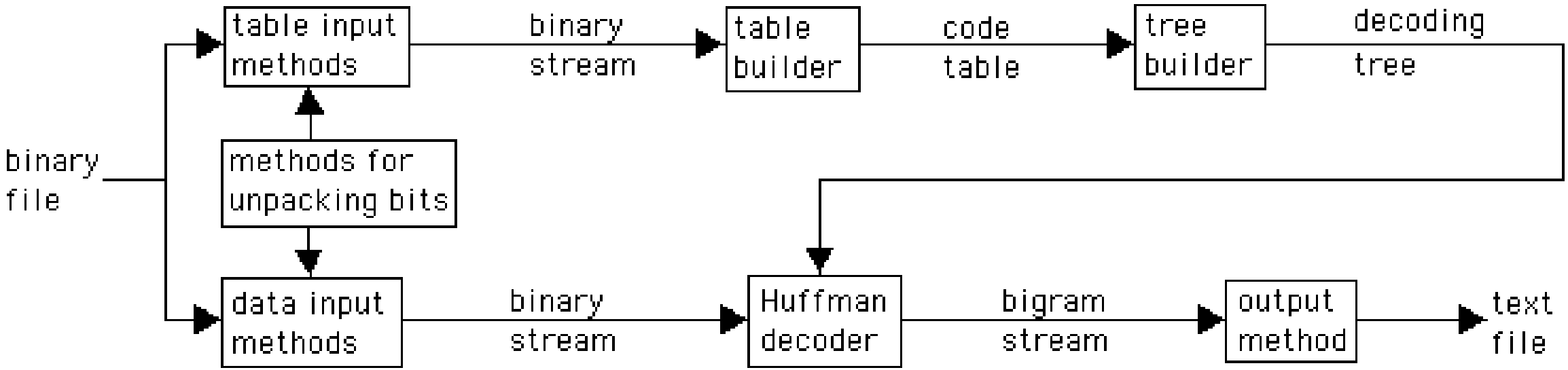


table input methods
binary stream
table builder
code table
tree builder
decoding tree
binary file
methods for unpacking bits
data input methods
binary stream
Huffman decoder
bigram stream
output method
text file

Application of Huffman Encoding

- Huffman is widely used in all the mainstream compression formats that you might encounter - from GZIP, PKZIP (winzip etc) and BZIP2, to image formats such as JPEG and PNG.
- All compression schemes have pathological data-sets that cannot be meaningfully compressed; the archive formats I listed above simply 'store' such files uncompressed when they are encountered.
- Newer arithmetic and range coding schemes are often avoided because of patent issues, meaning Huffman remains the work-horse of the compression industry.

JPEG

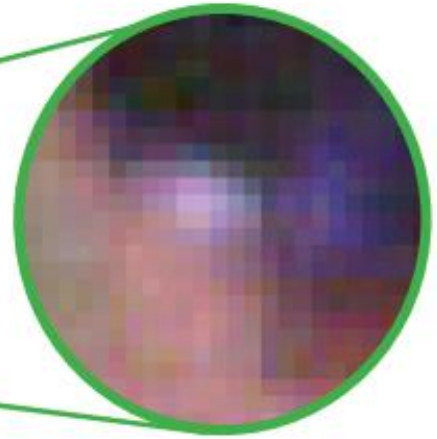
SECTION 5-1

What is JPEG Compression?

- **JPEG Compression** is the name given to an algorithm developed by the Joint Photographic Experts Group whose purpose is to minimize the file size of photographic image files. JPEG compression is a powerful tool, and with great power comes great responsibility.
- While JPEG compression can help you greatly reduce the size of an image file, it can also compromise the quality of an image - and if you aren't careful, there may not be any recovery.
- It is for this reason that we recommend saving your images in a lossless format such as TIFF.

JPEG Compression

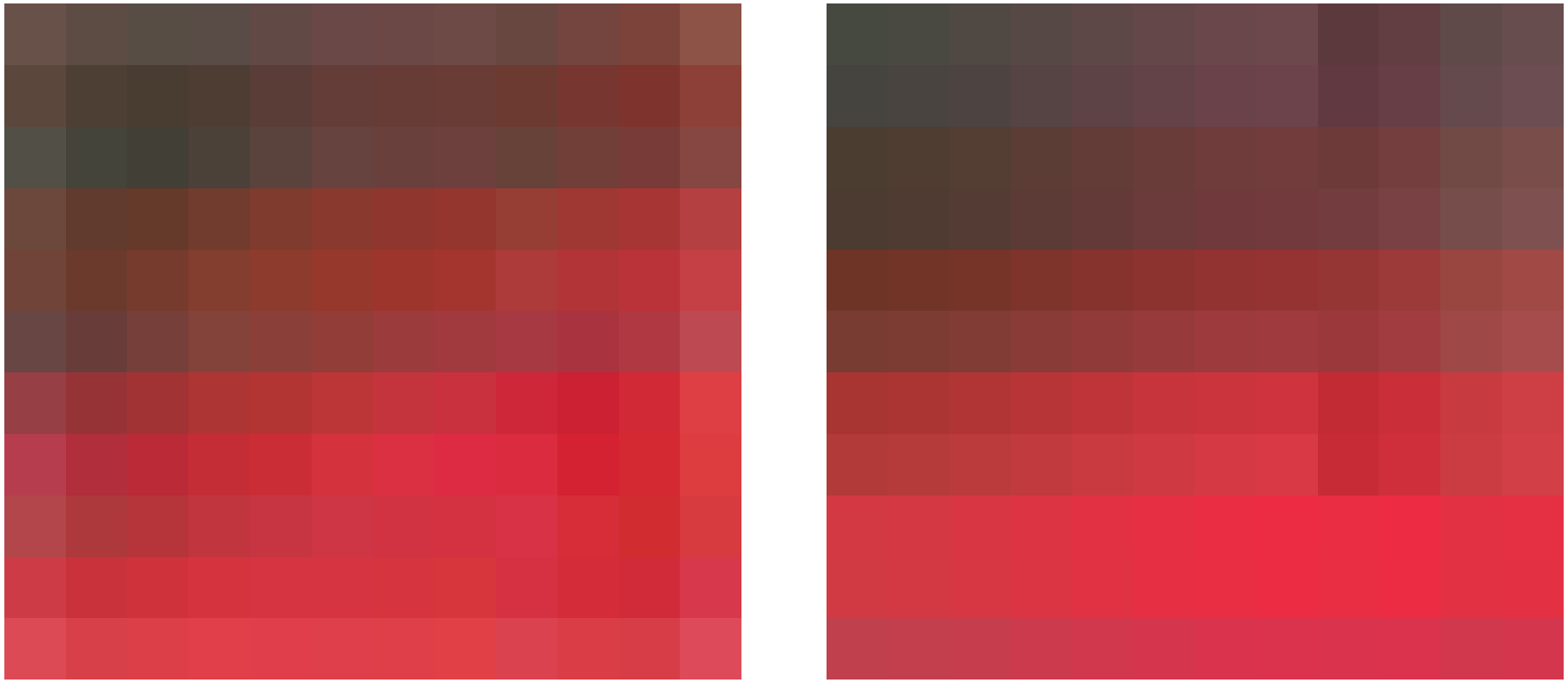
- Keep in mind that for the purposes of discussion, we are talking about **JPEG Compression**, which is not to be confused with the .JPG file format. While all .JPG files are indeed JPEG compressed, JPEG Compression can be used in many other file formats, including EPS, PDF, and even TIFF files.
- In order to understand JPEG compression better, it helps to understand how computers represent photographic images. An image file can be thought of as a grid of individual blocks called pixels. Each pixel has its own color value, and the larger the image, the more pixels.
- The more pixels, the larger the resulting file will be.



Photographic images are made up of individual blocks called pixels.

JPEG

- In order to reproduce the image once the file has been closed, the unique color of every pixel must be recorded. JPEG compression attempts to create patterns in the color values in order to reduce the amount of data that needs to be recorded, thereby reducing the file size.
- In order to create these patterns, some color values are approximated to match those of nearby pixels. You can almost think of it as “rounding off” the color values.

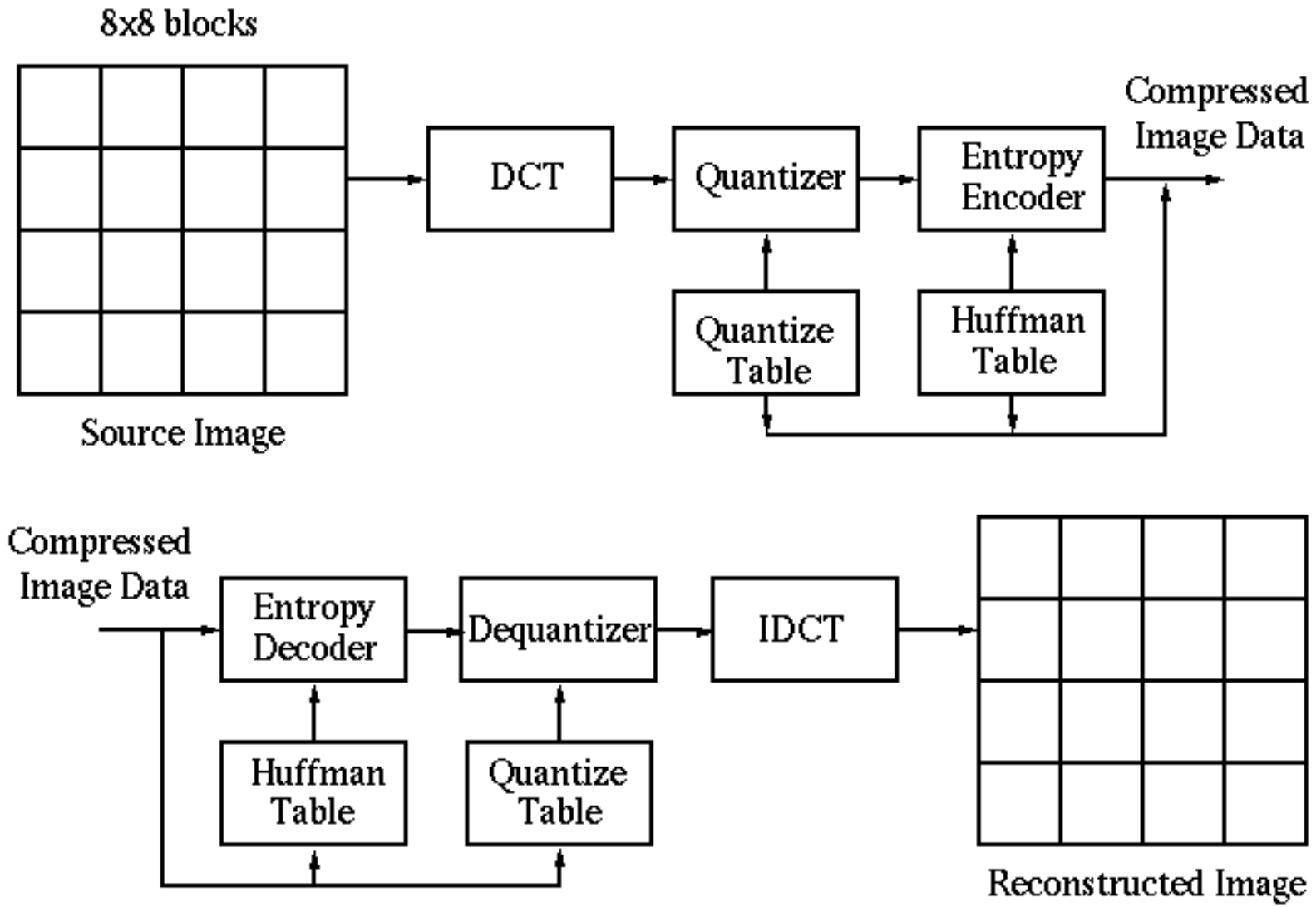


A sample of pixels pre-compression (left) and post-compression (right). Notice there are fewer individual colors and larger groupings of similar colors in the example on the right.

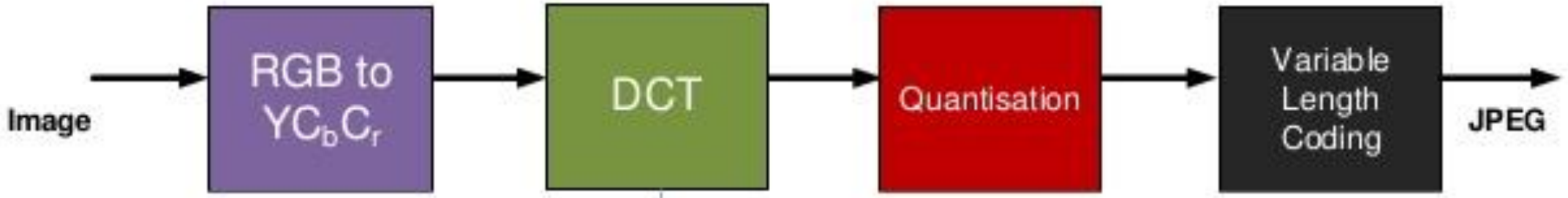
JPEG

- What does this mean for image quality? Fundamentally, JPEG compression tends to even out transitional colors: the points in an image where one color becomes a different color. This results in pixellization, where the image takes on a blocky appearance.
- Moreover, the JPEG Compression algorithm is applied every time you save your file, meaning that any damage being done to your image is cumulative and, in many cases, irreversible.

Clockwise from upper left: Progressively worsening compression artifacts caused by multiple saves of the same image.



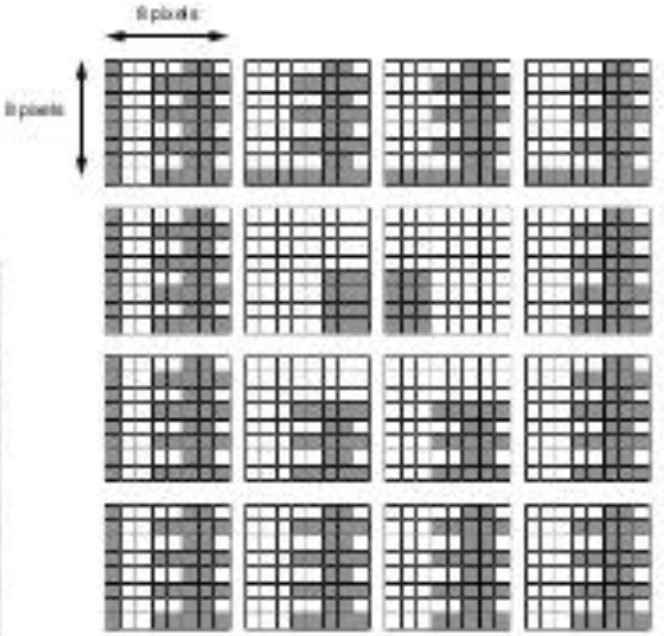
Discrete Cosine Transform (DCT)



$$F(u,v) = \frac{1}{4} C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

where $C(z) = \frac{1}{\sqrt{2}}$ if $z = 0$

or $= 1$ if $z \neq 0$

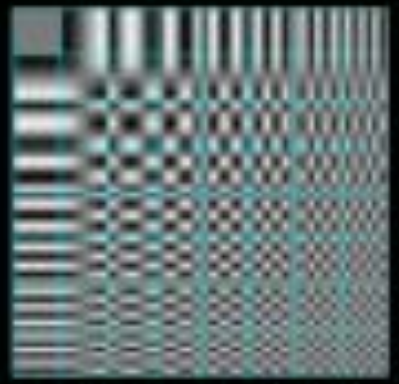


Link

8x8-Pixel-Block
aus RGB-Bild

RGB-Bild, konvertiert
in YUV

kanalweise
DCT-Transformation



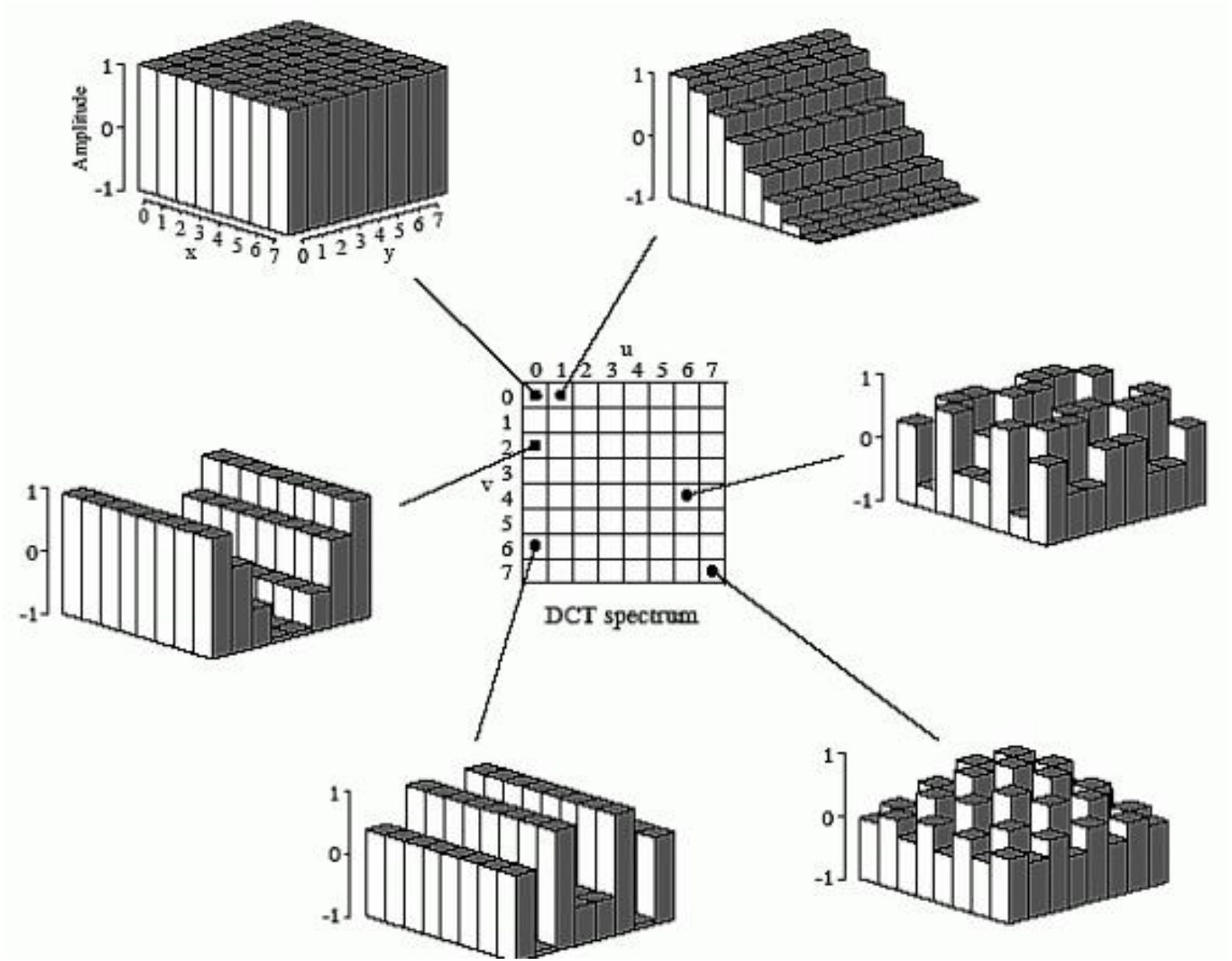
DCT-Basisfunktion

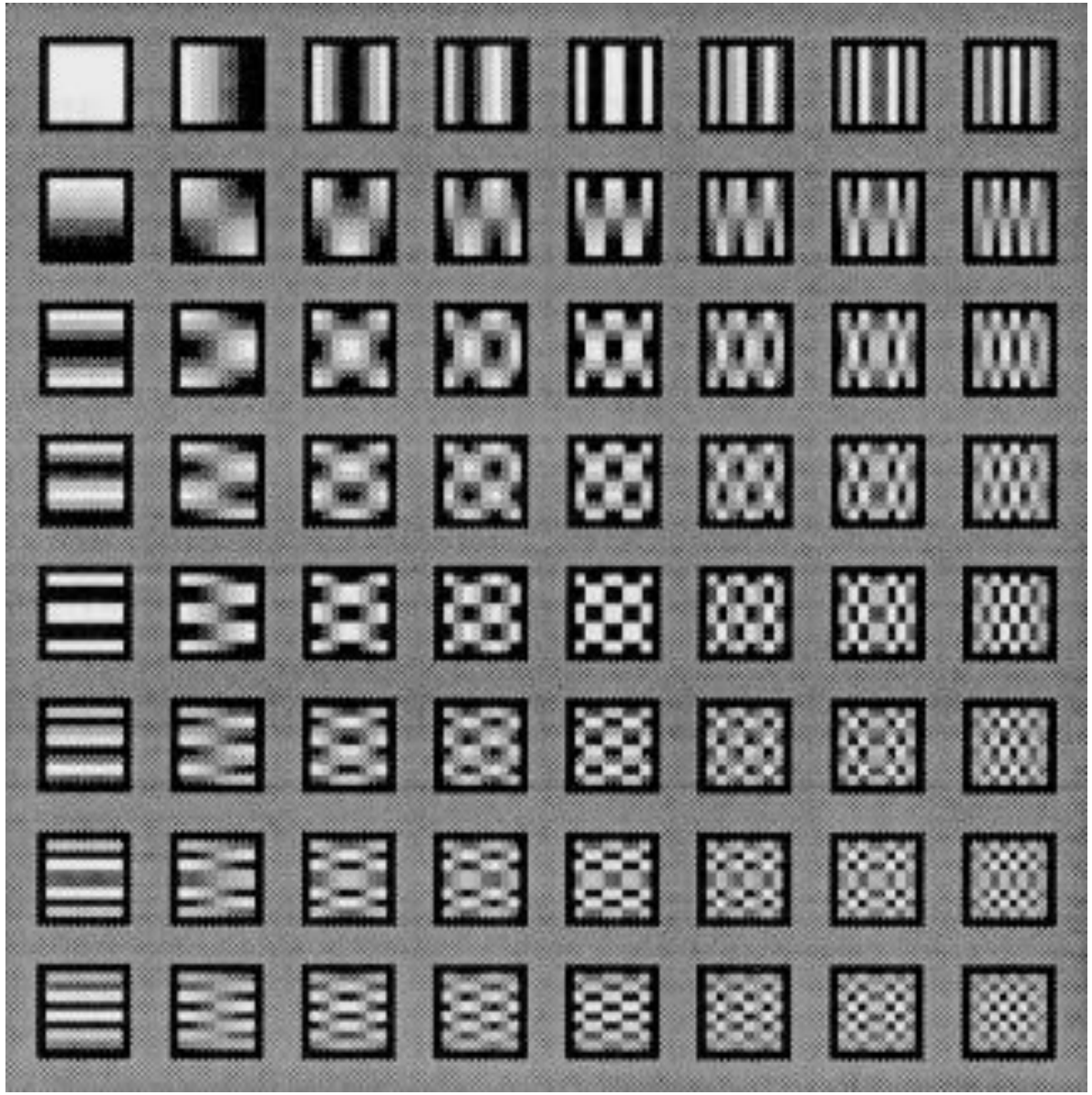
12	02	-9	-5	-1	1	-1	1
33	-9	-3	12	12	1	-3	2
10	12	10	10	5	3	3	1
54	20	9	10	-5	-1	5	1
-12	2	4	-3	-2	-2	1	1
10	3	-3	-5	-7	2	-2	2
25	10	3	4	6	-6	9	2
-9	3	2	-2	-1	-2	-2	-1

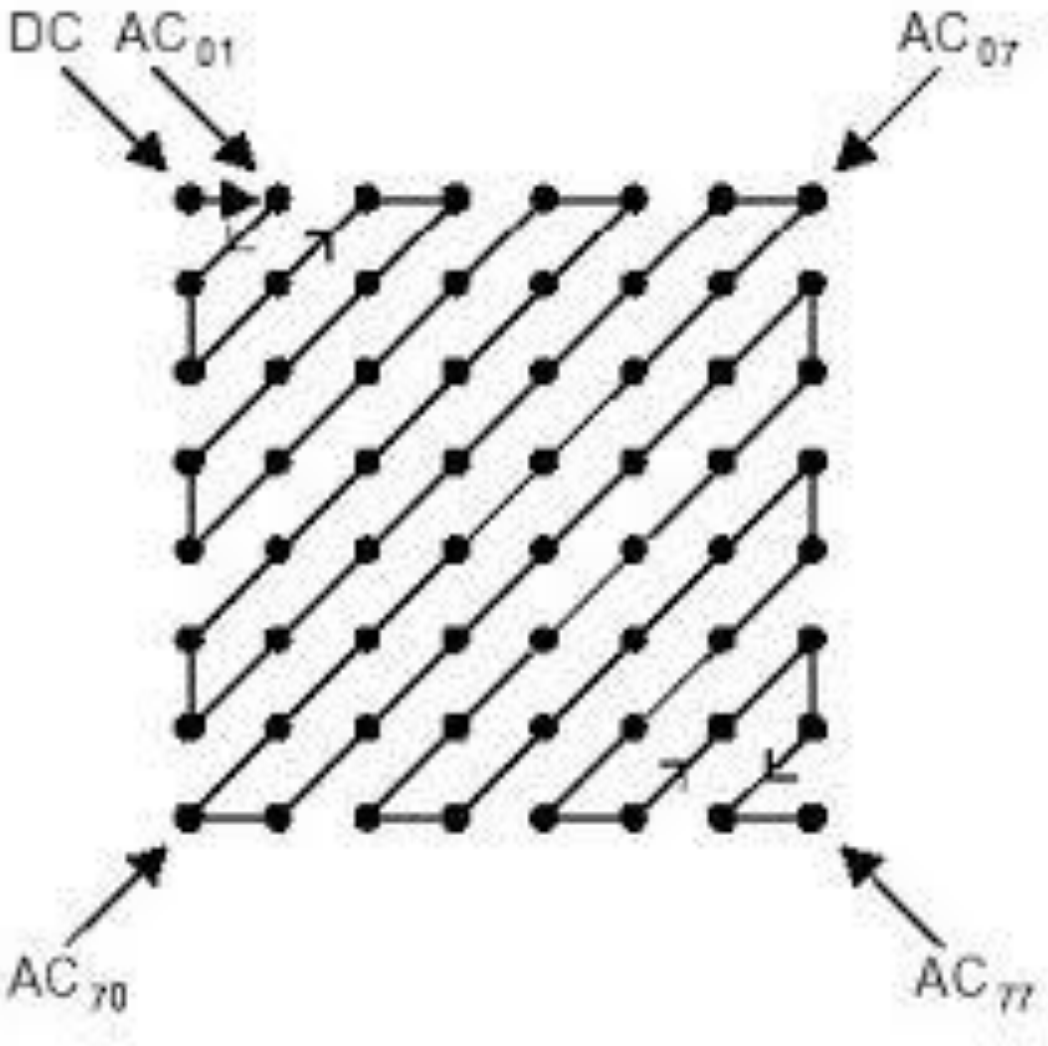
Koeffizientenmatrix
für die Helligkeit (Y)

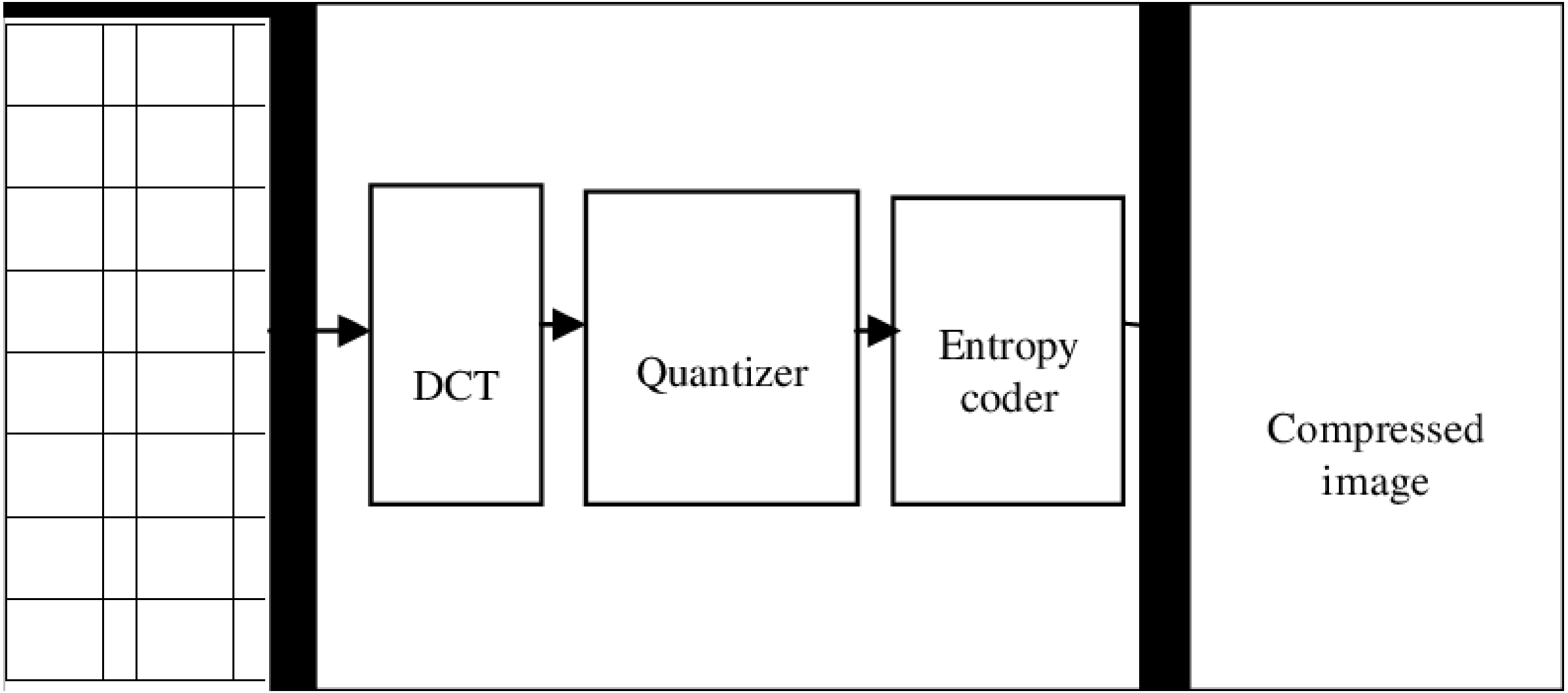
139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

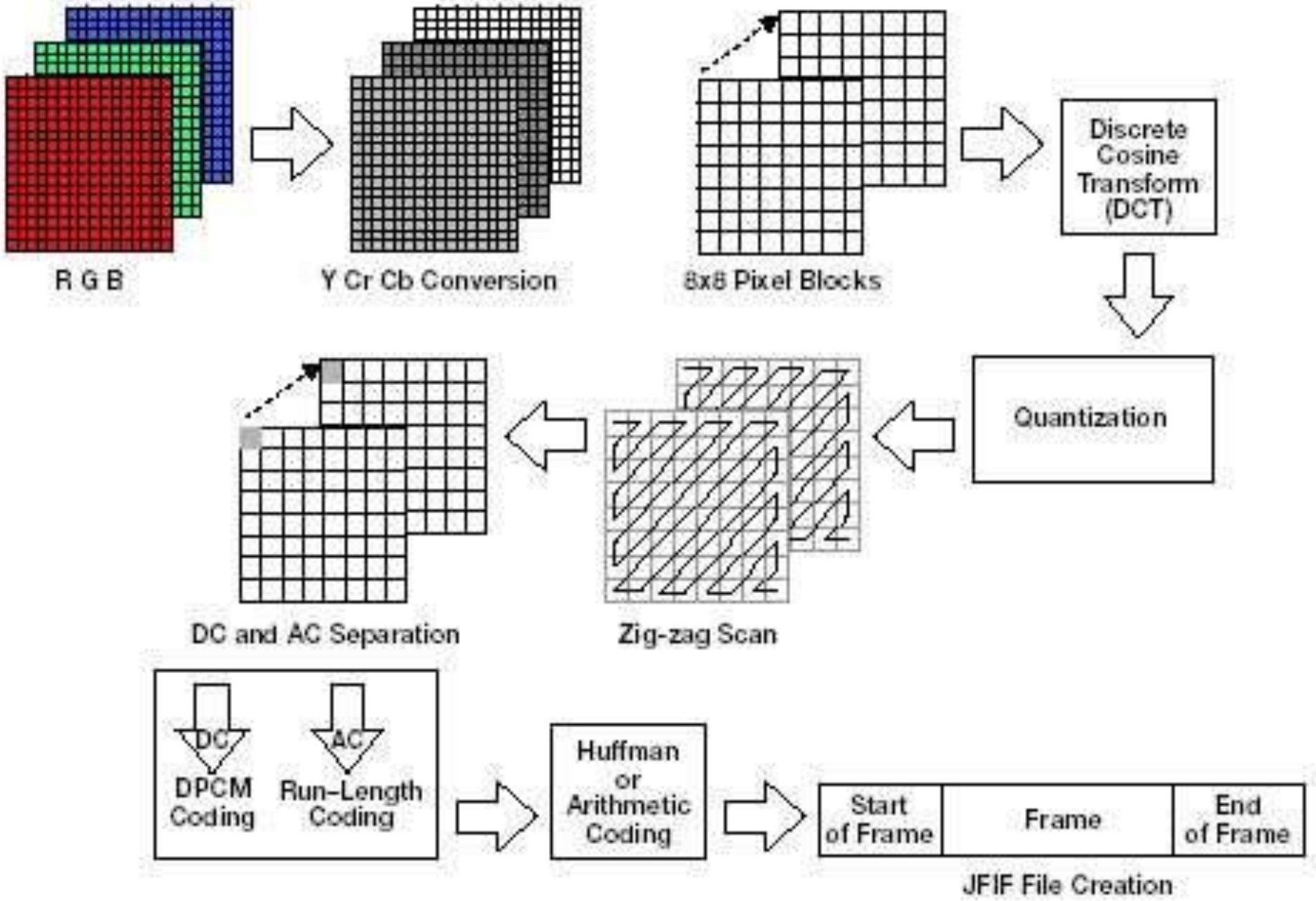
235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4


Amplitude
1
0
-1
0 1 2 3 4 5 6 7
x
0 1 2 3 4 5 6 7
y
u
v
DCT spectrum




DC
AC01
AC07
AC70
AC77





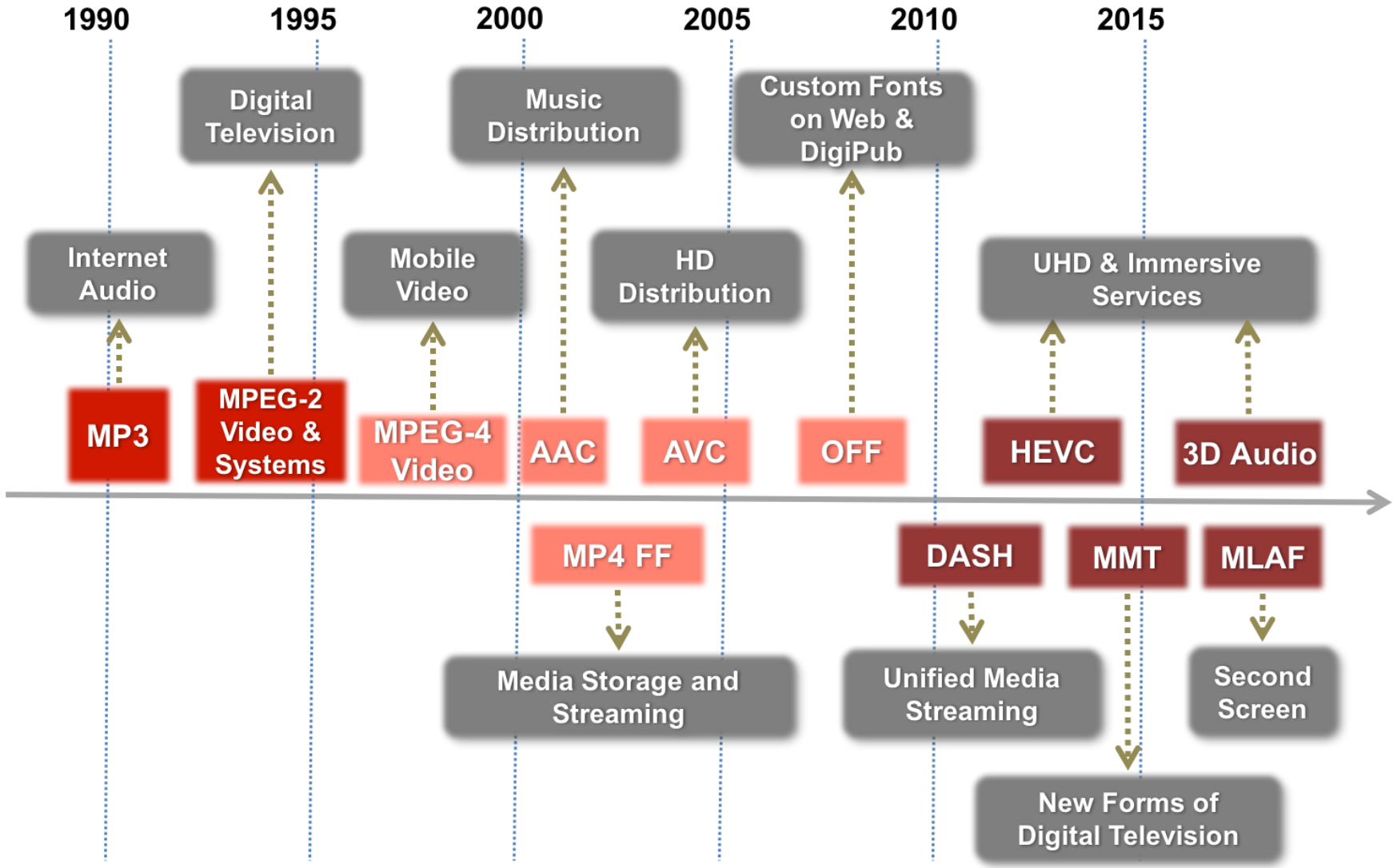
Video Compression

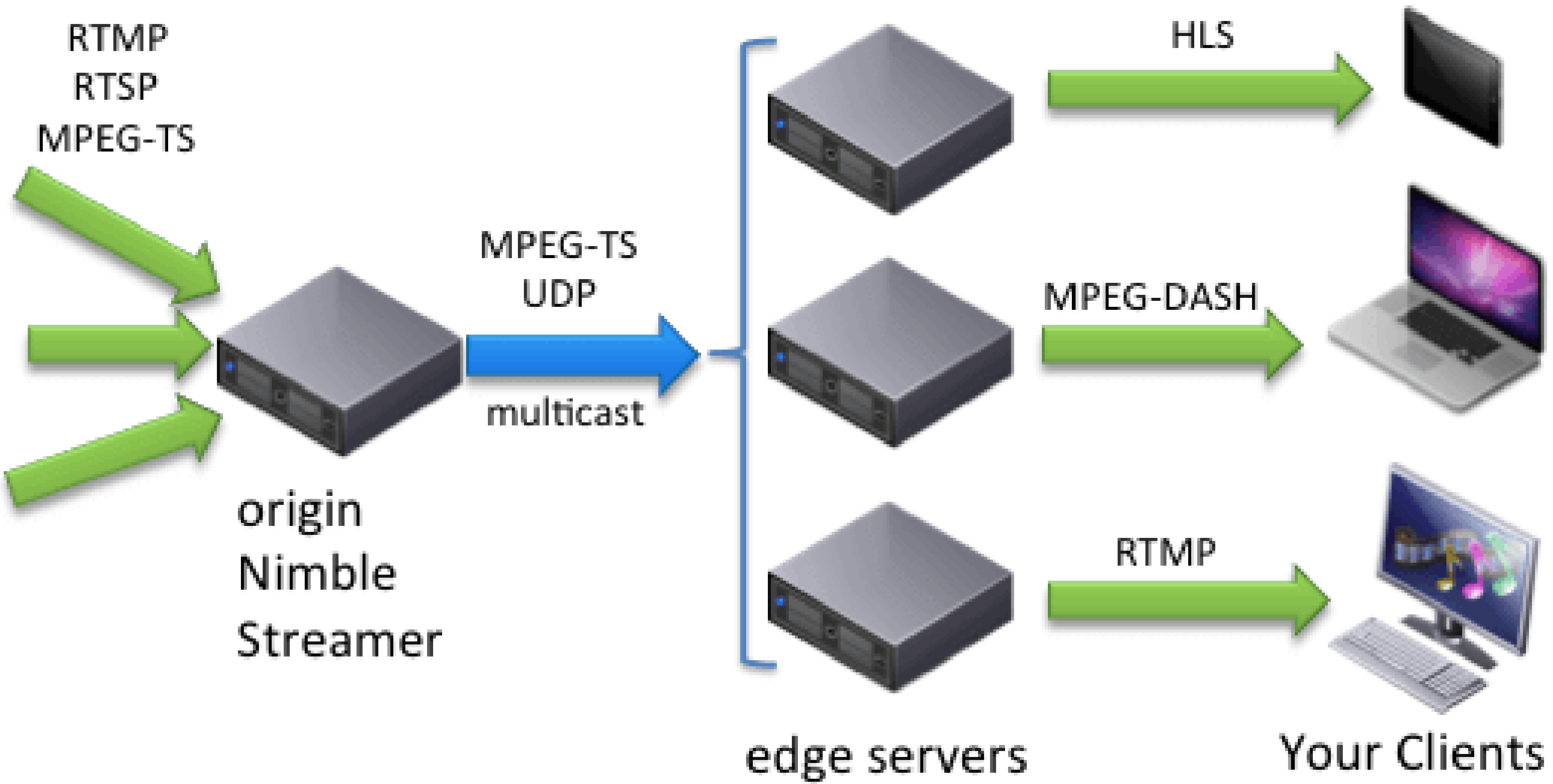
LECTURE 4

MPEG

SECTION 5-1

MPEG Standards




RTMP
RTSP
MPEG-TS
origin
Nimble
Streamer
MPEG-TS
UDP
multicast
HLS
MPEG-DASH
RTMP
edge servers
Your Clients

Need of Video Compression

- **Example:** HDTV broadcast has resolution of 1920*1080 at 30 fps using 8 bits to represent primary colors.
  - This leads to a total of 1.5 Gbps data rate
- But, Channel B.W. is only 6MHz that supports around 19.2 Mbps data rate only.
  - But, channel also has to support audio, and other data.
  - Effective data rate reduced to only 18 mbps

Compression Ratio

Therefore, Compression required is

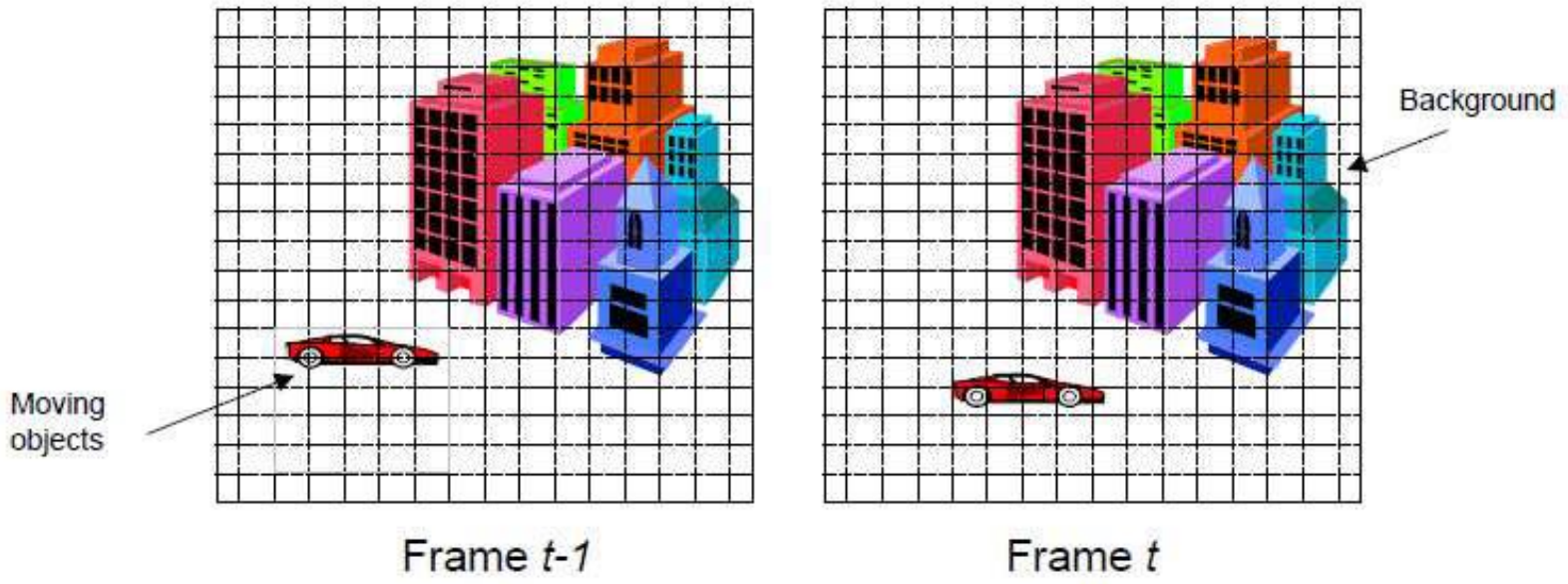
Compression = 1.5 Gbps / 18mbps = 83

Hence, Compression ratio required is 83: 1

Basics of Video Compression

- Video is made of multiple images called **frames**.
- Images are having highly redundant data i.e. adjacent pixels are highly correlated.
- JPEG compresses images.

Characteristic of Video



- Adjacent frames are similar and changes are due to object motion:

Temporal Correlation

Motion Estimation

Key idea

Predict a new frame from previous frame and specify the prediction error.

Prediction error can be coded using **image coding methods**. (e.g. JPEG)

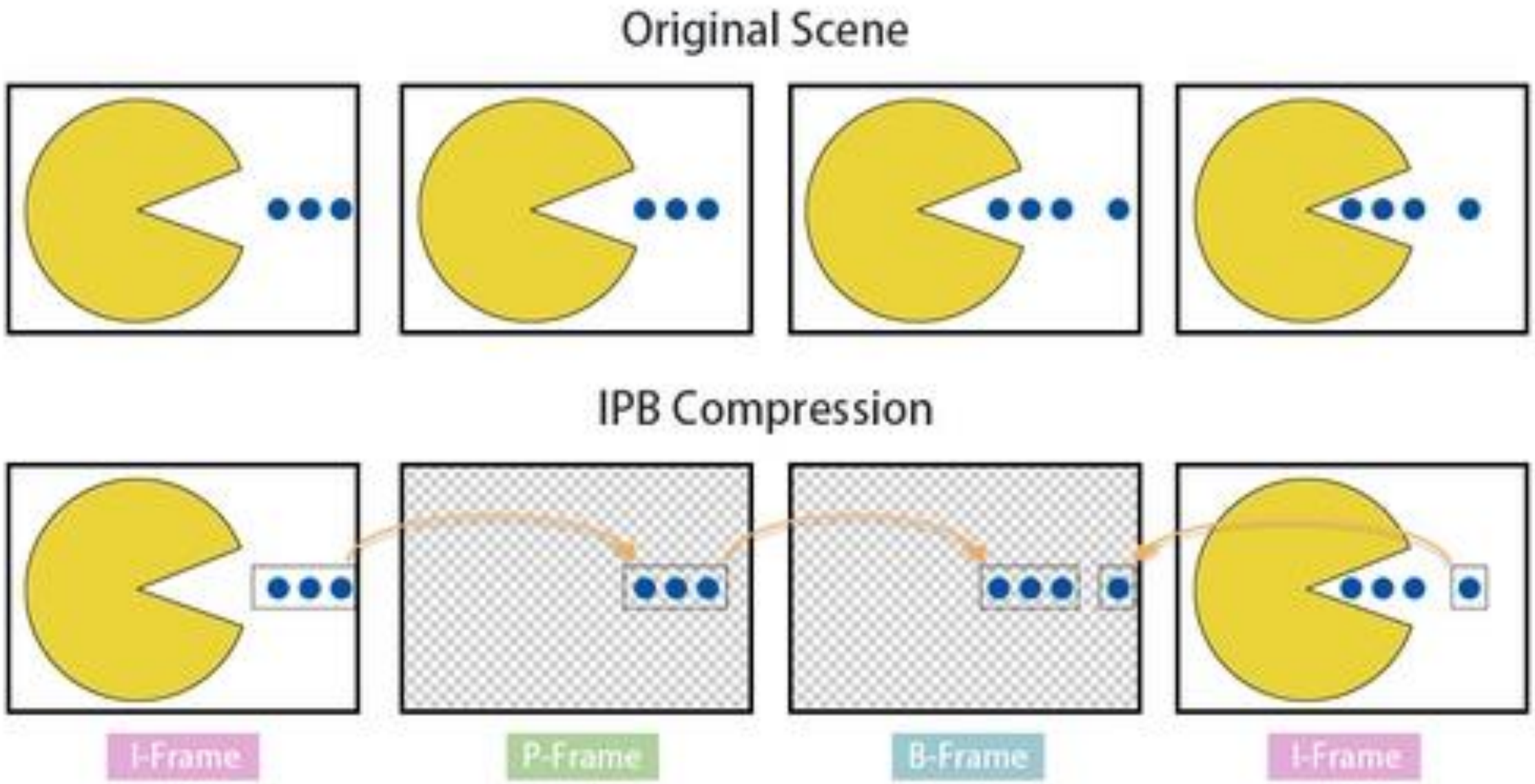
Predictions from past frame are known as **forward prediction**.

Prediction error can be coded with fewer bits

Regions those can’t be predicted well, are coded directly.

MPEG

- Previous of next frame is known as **reference image**.
- Video compression makes use of **MOTION COMPENSATION** to predict a frame from the previous and/ or next frame.
- **Motion vectors** that describes the transformation from one image to another that are applied to the target and synthesized to produce next image.


Original Scene
IPB Compression
I-Frame
P-Frame
B-Frame
I-Frame

Key terms

Macro Blocks: Compression methods works on a block of 16*16 pixels called macro blocks.

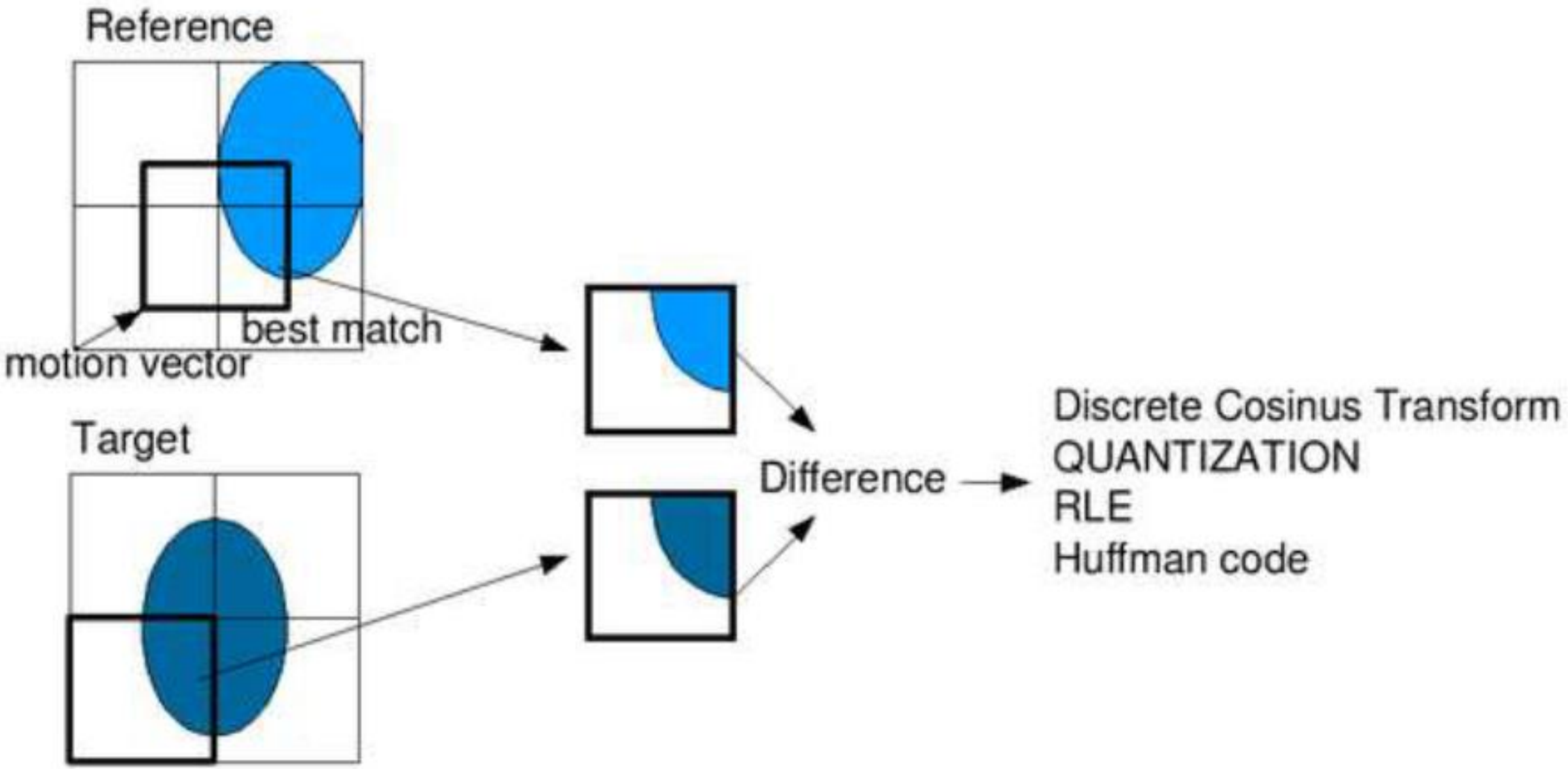
I (intra) Frames

- Independent frames
- Coded without reference to other frames

Key Terms

P (predictive) Frame

- Not Independent
- These frames are **predicted** from a past frame (I or P)
- Coded by a forward **predictive coding**
  - Current macro block is predicted from similar macro block in previous I or P frame. And the difference between macro blocks is coded



Key Terms

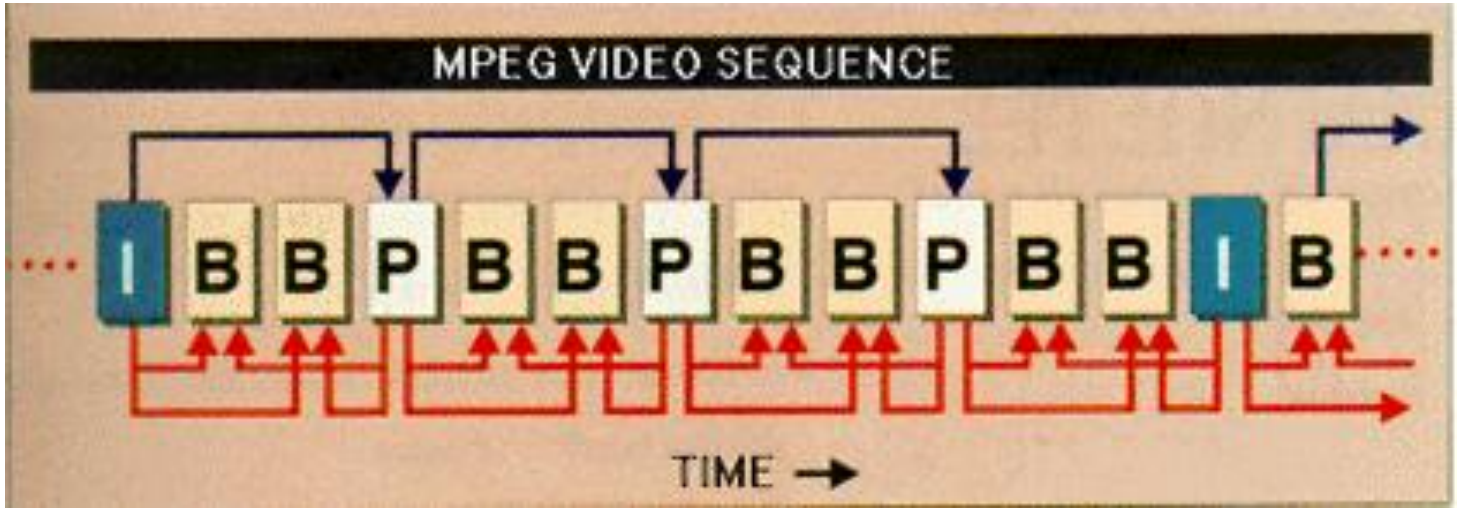
MPEG introduced a new **B frame**

B frame (bi-directional predictive-coded)

- Due to unexpected movement in real scenes, the target macro block may not have a good matching in previous frame.
- Therefore, B frame is coded with reference to both previous and future reference frames (either I or P)

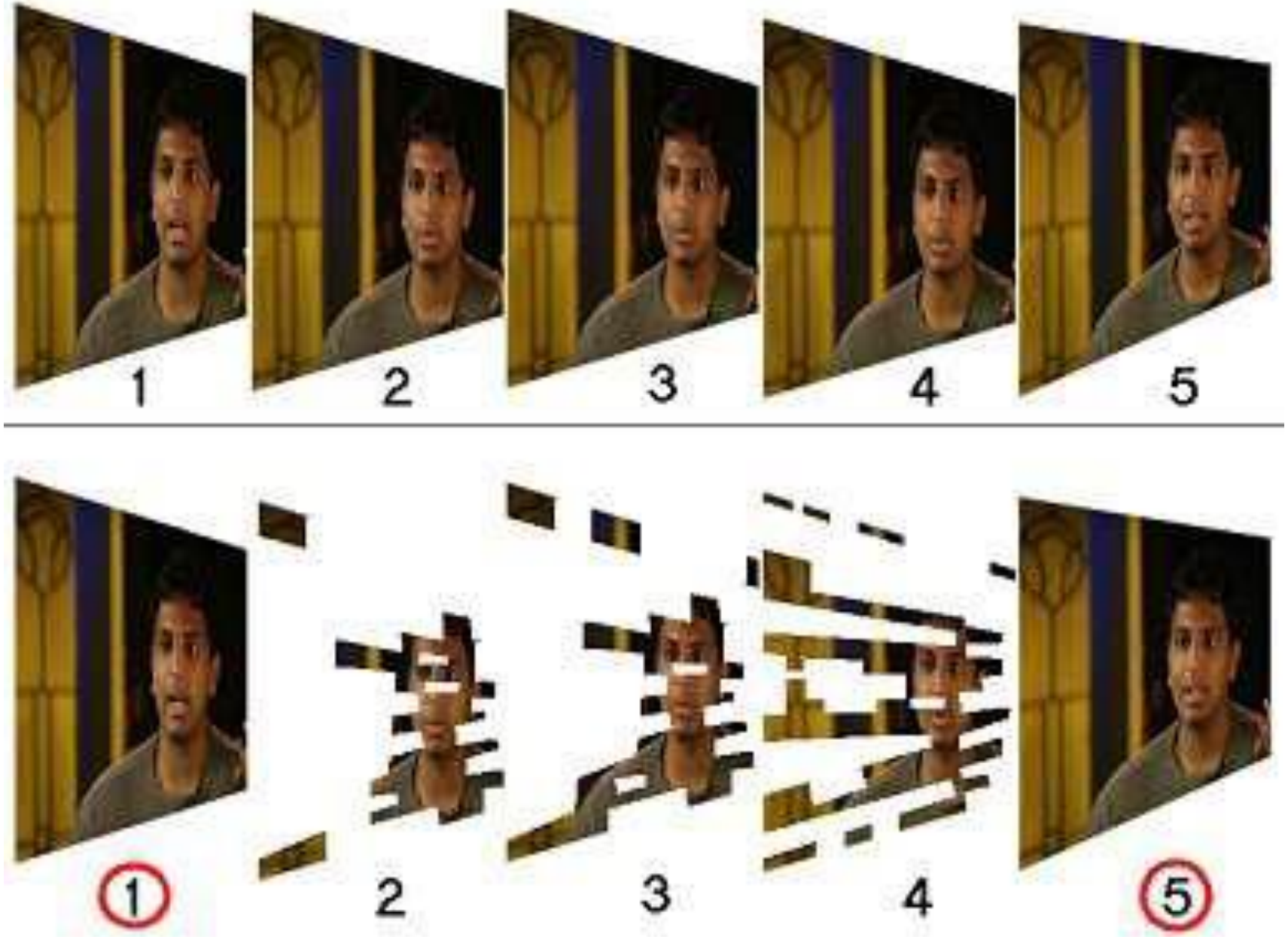
Frame Sequence

- Actual frame sequence is determined by encoder and is specified in video frame header.

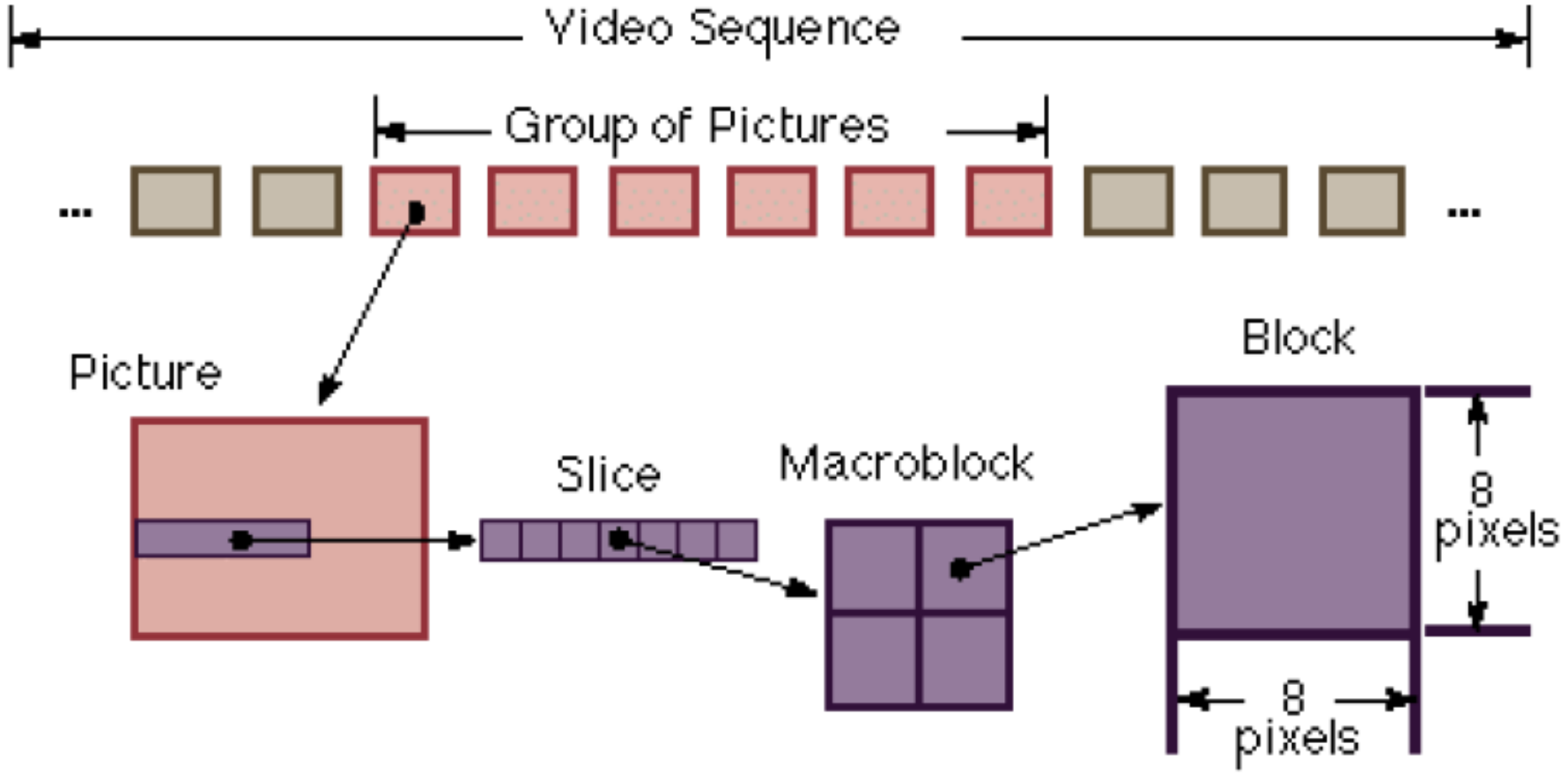


Display order: I B B P B B P B B I
Decoding Order: I P B B P B B I B B

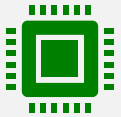
NOTE: Decoder can't work with B frame without its succeeding P or I frame


1
2
3
4
5
1
2
3
4
5

Video Stream Data Hierarchy



Four Video Compression Techniques



1. Pre-processing

2. Temporal Prediction

3. Motion Compensation

4. Quantization

Pre-processing

Filters out unnecessary information

- Information that is difficult to encode
- Not an important component of human visual perception

Temporal Prediction

Uses the mathematical algorithm Discrete Cosine Transform (DCT) to:

- Divide each frame into 8X8 blocks of pixels
- Reorganize residual differences between frames
- Encode each block separately

Only Moving Areas Have to Be Coded

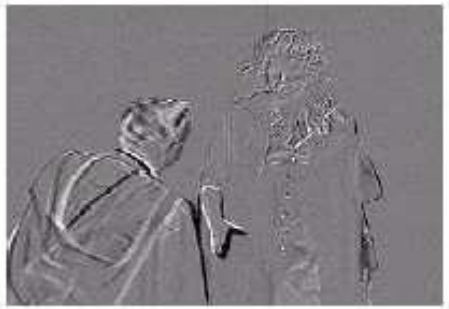
new picture

-

previous picture

=

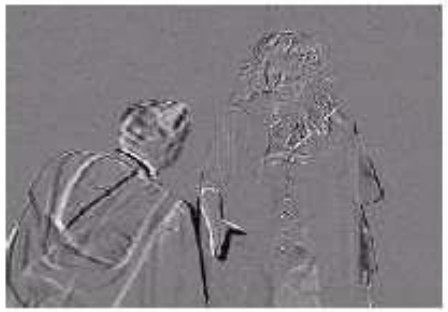
difference



Encoder

Decoder

difference



+

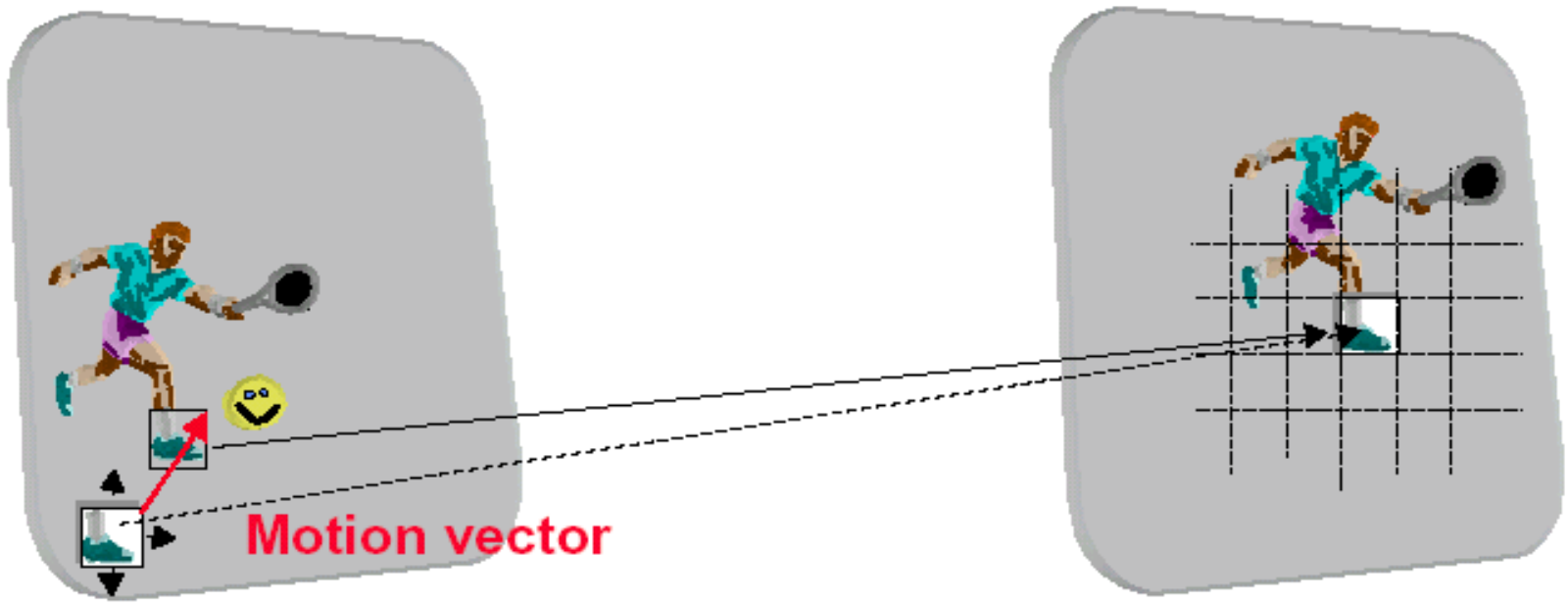
previous picture

=

new picture

Motion Compensation

- Try to match each block in the actual picture to content in the previous picture. Matching is made by shifting each of the 8 x 8 blocks of the two successive pictures pixel by pixel each direction -> Motion vector
- Substract the two blocks -> Difference block
- Transmit the motion vector and the difference block



Quantization

- Lossy Compression
- Compressed range of value by a single quantum value
- Quantization Matrix is designed to provide more resolution to perceivable frequency
- Set much of the unnecessary elements to Zero
- Uses zig zag ordering and run length encoding.