Multimedia Content

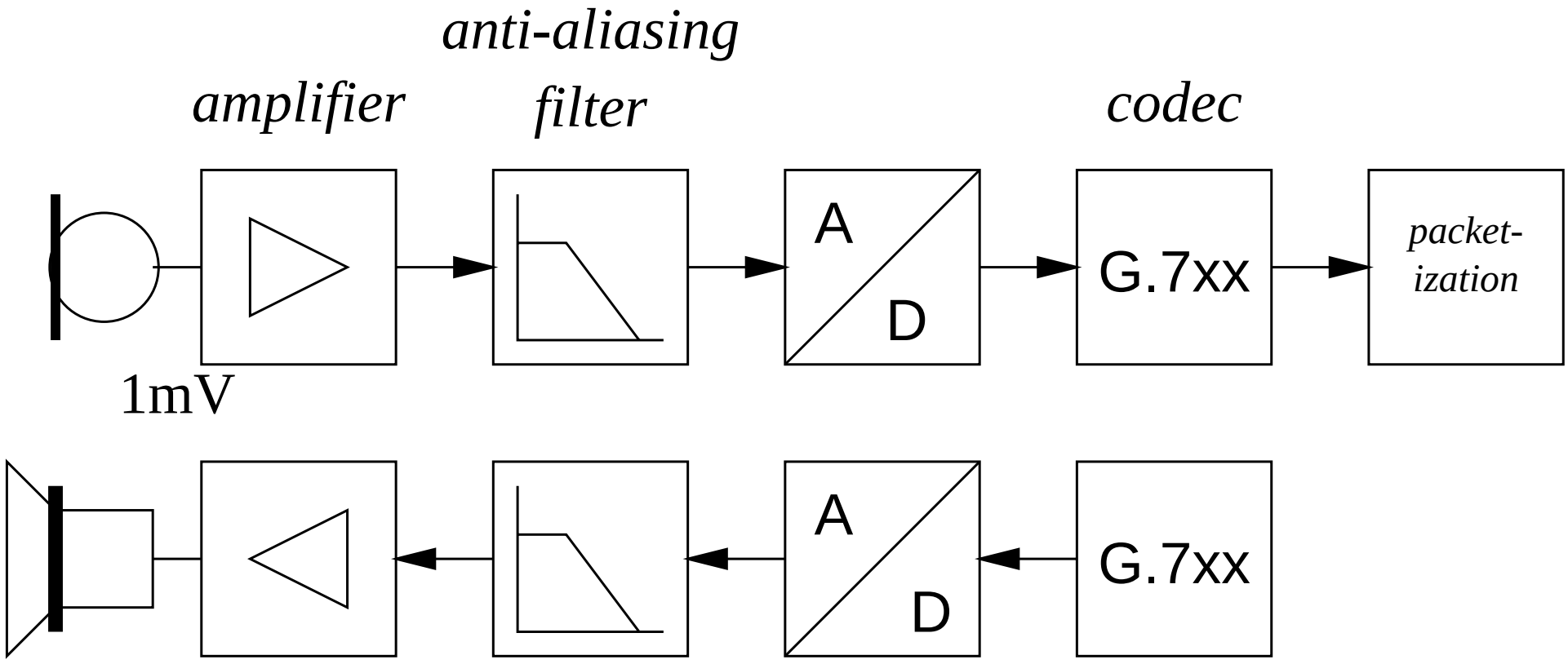
Multimedia review

- digital audio and video
- encodings
- quality evaluation

References

- J. Bellamy, *Digital Telephony*, 2nd ed., Wiley, 1991.
- N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice Hall.
- A. S. Tanenbaum, *Computer Networks*. Upper Saddle River, New Jersey: Prentice-Hall, 3rd ed., 1996. (Chapter 7) [Figures drawn from this book]
- R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*. Upper Saddle River, New Jersey: Prentice-Hall, 1995.
- O. Hersent, D. Gurle and J.P. Petit, *IP Telephony*, Addison-Wesley, 2000.
- L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, 1978.

Digital sound



dB and sound levels

- SNR (decibel, dB) $= 10 \log_{10}(\sigma_x^2/\sigma_y^2)$
- for amplitudes: dB $= 20 \log_{10} x/y$
- relative to 1 mW: dBm
- also: dynamic range
- sound pressure level (relative to $10^{-12}\mathrm{W/m^2}$)

Colt .45 at 25 feet	145 dB	factory	80 dB
threshold of pain	120–130 dB	conversation	60-65 dB
underground train	115 dB	office	50 dB
average home stereo	90 dB	quiet whisper (5')	25 dB
		hearing threshold	0 dB

- FM ≈ 40 dB, TV < 40 dB, audio CD 96 dB

Some sound physics

- $T = 1/f$, $v = \lambda f$ = 333 m/s (depends on temperature)
- amplitude, phase and frequency
- harmonics (octave) and formants:

sound	F_1	F_2	F_3
beet	270	2290	3010
bit	390	1990	2550
boot	300	870	2240
bat	660	1720	2410

- unvoiced (s, f, sh): wide spectrum (“white noise”)
- greatest sensitivity around 1-1.5 kHz: 10 kHz tone must be 8 dB louder
- high-frequency hearing sensitivity decreases with age

Filters

Finite impulse response (FIR): $y(n) = x(n) + \alpha_1 x(n-1) + \alpha_2 x(n-2) + \ldots$

Infinite impulse response (IIR): $y(n) = \beta_1 y(n-1) + \beta_2 y(n-2) + \ldots + \alpha_0 x(n)$

IIR filters have sharper cut-offs

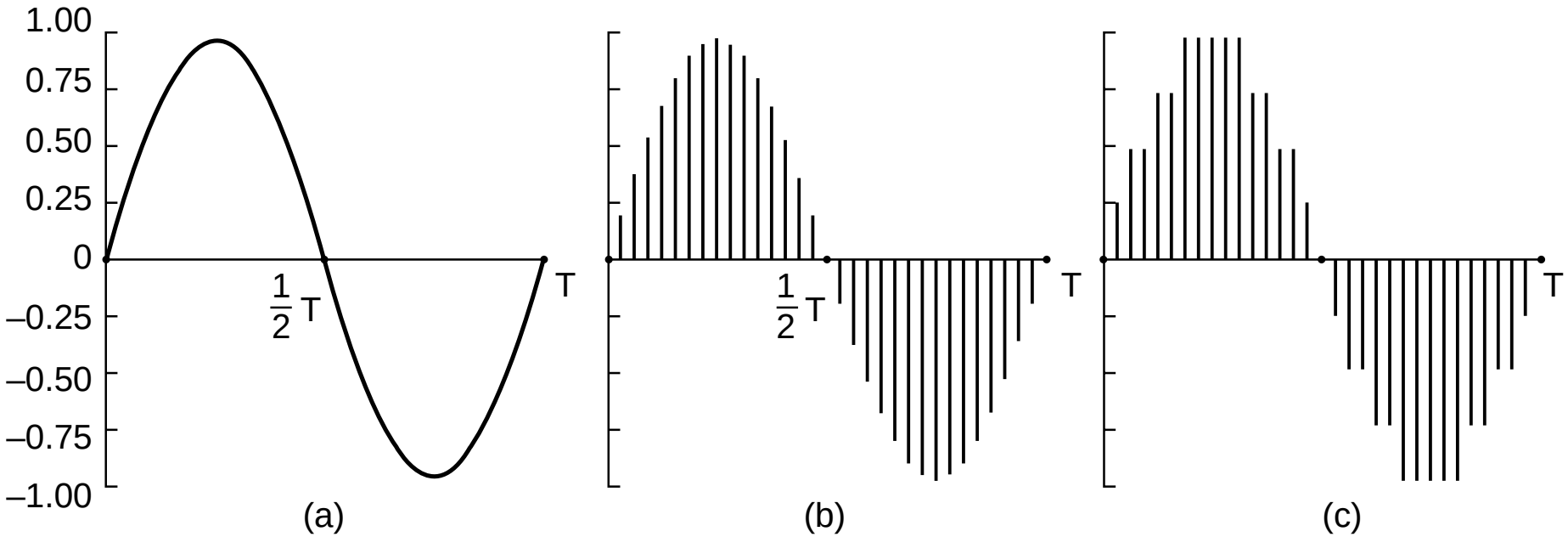
Digital audio

- sample each audio channel and quantize ➠ pulse-code modulation (PCM)
- Nyquist bound: need to sample at twice (+ ϵ) the maximum signal frequency
- analog telephony: 300 Hz – 3400 Hz ➠ 8 kHz sampling $\longrightarrow$ 8 bits/sample, 64 kb/s
- FM radio: 15 kHz
- audio CD: 44,100 Hz sampling, 16 bits/sample (based on video equipment used for early recordings)
- more bits ➠ more dynamic range, lower distortion
- audio highly redundant ➠ compression
- almost all codecs fixed rate

Audio coding

application	frequency	sampling	AD/DA bits	application
telephone	300-3400 Hz	8 kHz	12–13	PSTN
wide band	50-7000 Hz	16 kHz	14–15	conferencing
high-quality	30-15000 Hz	32 kHz	16	FM, TV
	20-20000 Hz	44.1 kHz	16	CD
	10-22000 Hz	48 kHz	≤ 24	pro-audio

Digital audio: sampling



distortion: signal-to-(quantization) noise ratio

Digital audio: compression

Alternatives for compression:

- companding: non-linear quantization ➠ μ-law (G.711)
- waveform: exploit statistical correlation between samples
- model: model voice, extract parameters (e.g., pitch)
- subband: split signal into bands (e.g., 32) and code individually ➠ MPEG audio coding

Newer codings: make use of *masking properties* of human ear

Judging a codec

- bitrate
- quality
- delay: algorithmic delay, processing
- robustness to loss
- complexity: MIPS, floating vs. fixed point, encode vs. decode
- tandem performance
- can the codec be *embedded*?
- non-speech performance: music, voiceband data, fax, tones, . . .

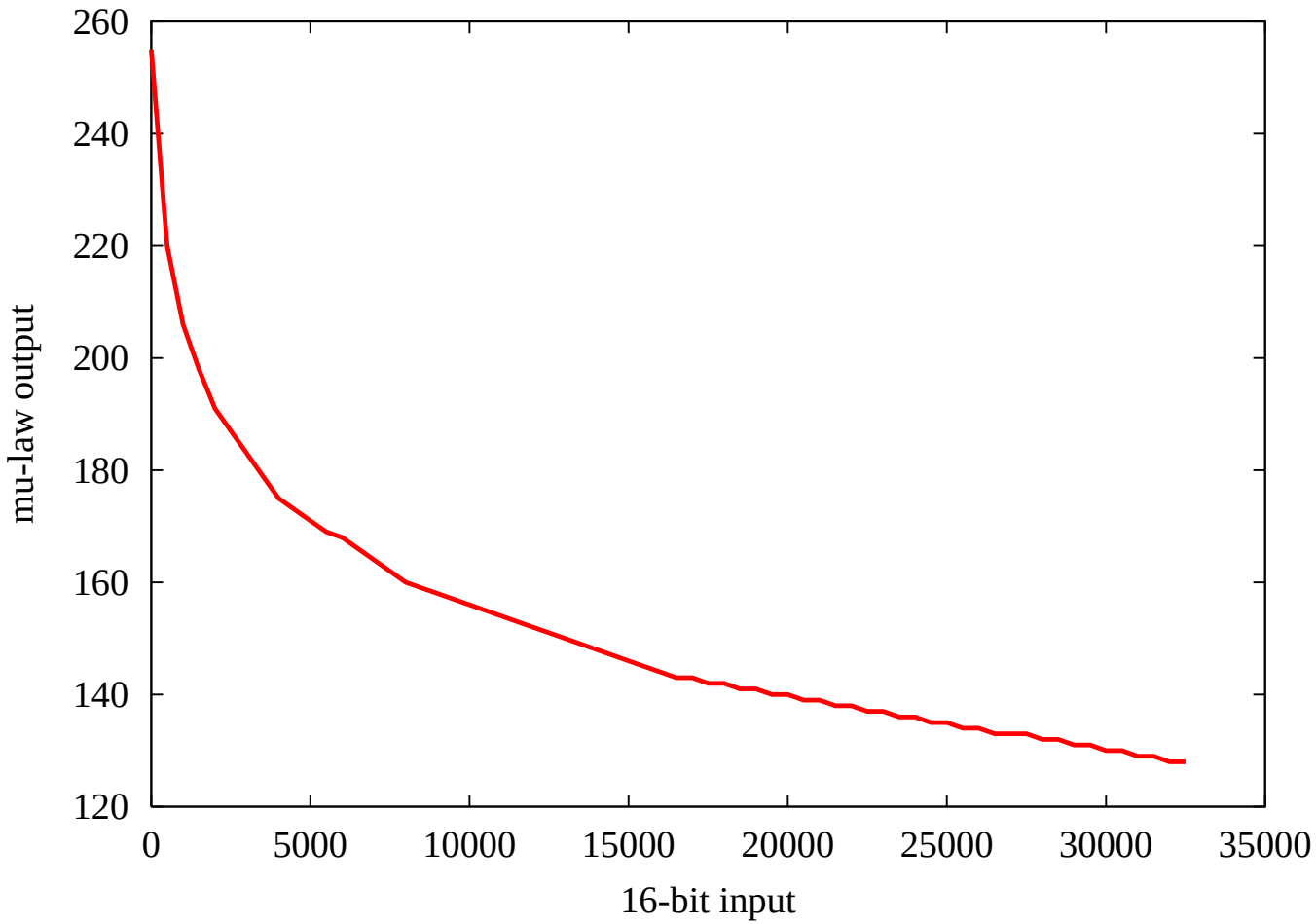
Quality metrics

- speech vs. music
- communications vs. *toll quality*
- mean opinion score (MOS) and degradation MOS

score	MOS	DMOS	
5	excellent	inaudible	no effort required
4	good, toll quality	audible, but not annoying	no appreciable effort
3	fair	slightly annoying	moderate effort
2	poor	annoying	considerable effort
1	bad	very annoying	no meaning

- diagnostic rhyme test (DRT) for low-rate codecs (96 pairs like “dune” vs. “tune”) – 90% = toll quality

Companding: μ-law for G.711 ("PCMU")



Also: A-law in Europe

Silence detection (VAD)

- detect silence based on energy, sound
- avoid transmitting silence during sentence pauses and/or other person talking
- hangover
- conferencing!
- comfort noise – white noise
- transmit update (4 byte) when things change

Silence model

- speech only, not for music
- strongly depends on VAD mechanism
- silence periods of 1.0-1.6 seconds
- roughly, exponentially distributed
- sensitive to background noise

Speech codecs

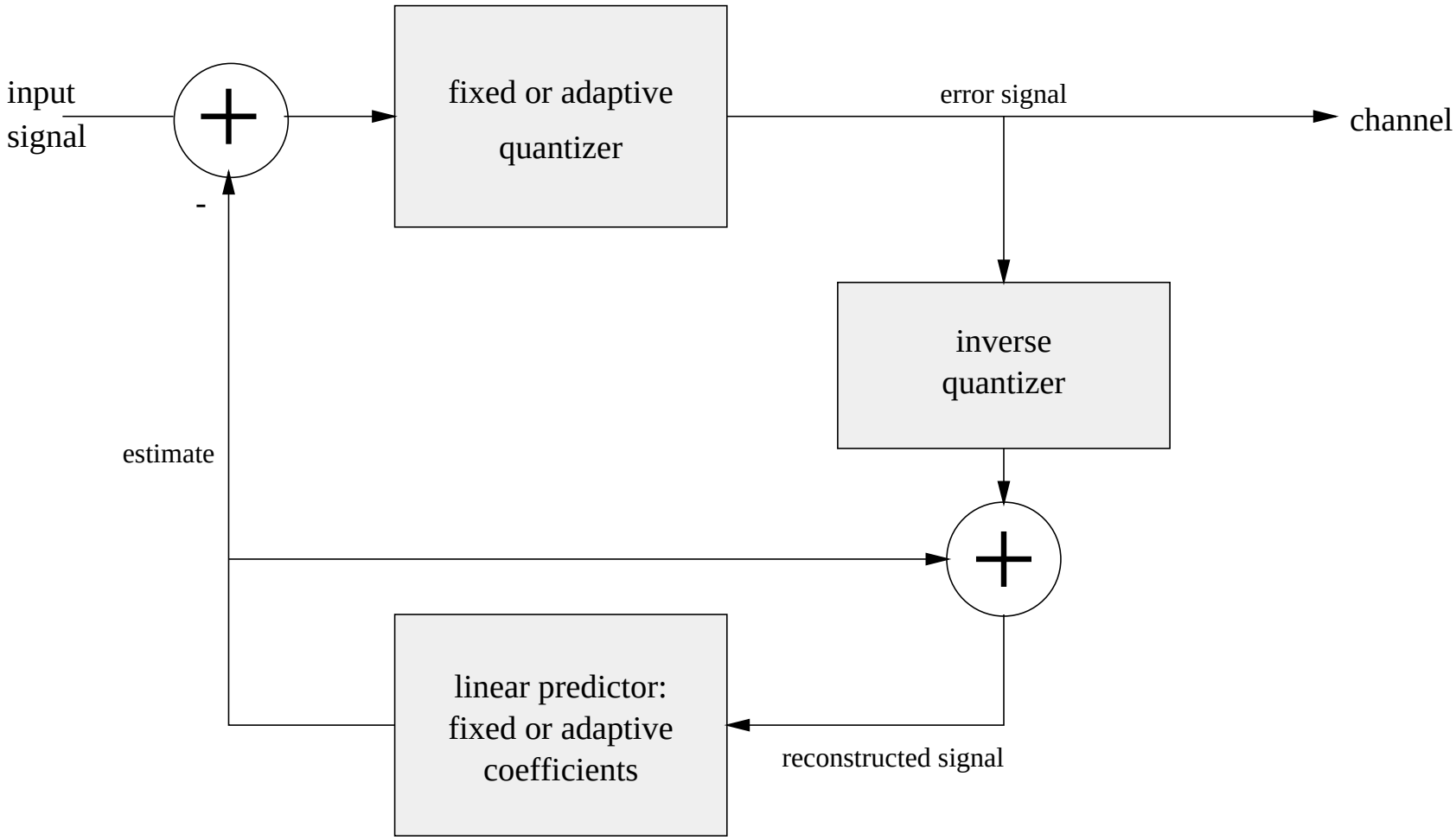
- waveform codecs exploit sample correlation: 24-32 kb/s
- linear predictive (vocoder) on *frames* of 10–30 ms (stationary): remove correlation $\longrightarrow$ error is white noise
- vector quantization
- hybrid, analysis-by-synthesis
- entropy coding: frequent values have shorter codes
- runlength coding

Digital audio: compression

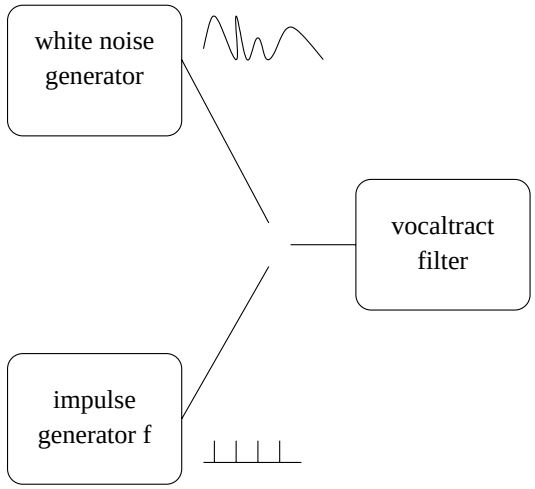
coding	kb/s	MOS	use
LPC-10	2.4	2.5	robotic, secure telephone
G.723.1	5.3/6.3		videotelephony (room for video)
G.729	8.0		mobile telephony
G.726	16-40		low-complexity (ADPCM)
GSM	13.0	3.54	European mobile phone
G.728	16.0	4.0	low-delay
DVI	32.0		toll-quality (Intel, Microsoft)
G.722	64.0		7 kHz codec (subband)
G.711	64.0	4.3	telephone (μ-law, A-law)
MPEG L3	56-128.0	N/A	CD stereo
16 bit/44.1 kHz	1411	compact disc	

See also http://www.cs.columbia.edu/~hgs/audio

(Adaptive) Differential Pulse Code Modulation



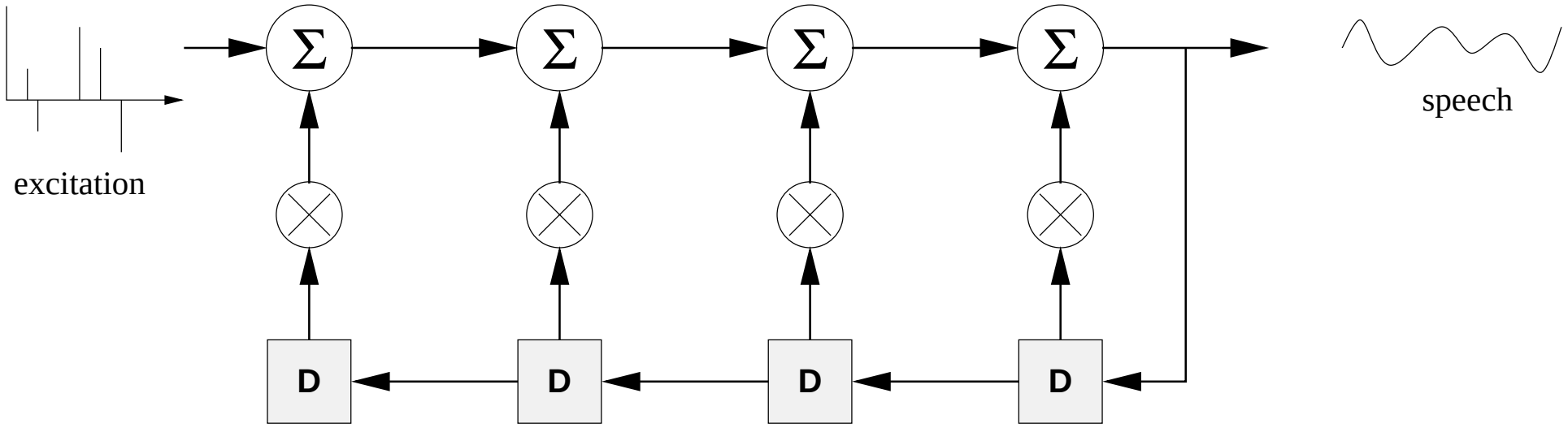
Linear predictive codec



E.g., LPC 10 at 2.4 kb/s:

Sampling rate	8 kHz
Frame length	180 samples = 22.5 ms
Linear predictive filter	10 coefficients = 42 bits
Pitch and voicing	7 bits
Gain information	5 bits

Linear predictive codec



linear (IIR) filter

infinite impulse response (IIR) vs. finite impulse response

G.723 audio codec

- analysis-by-synthesis codec
- 5.3 or 6.3 kb/s bit rate
- 30 ms frames with 7.5 ms look-ahead
- MOS of 3.98
- requires 40% of 100 MHz Pentium or 22 DSP MIPS, 16 kB code
- popular for videoconferencing with modems

G.729 audio codec

- 8 kb/s bit rate
- 10 ms frames with 5 ms look-ahead
- LD-CELP (low-delay code-excited linear prediction): filter coefficients, code book for excitations
- requires 25% of 100 MHz Pentium
- MOS of 3.7/3.2/2.8 with 0/3/5% packet loss
- deals with *frame erasure*
- Annex A: low-complexity; Annex B: VAD; Annex D: 6.4 kb/s; Annex E: 11.8 kb/s

Audio perceptual encoding

- ear = 24-26 overlapping bandpass filters (100 Hz to 5,000 Hz)
- maximum sensitivity between 1,000 and 5,000 Hz
- masked by stronger signal in close frequency proximity
- ISO MPEG-1 Layer I, II, III
- analysis filter bank
- AAC: 64 kb/s for mono for almost CD-quality

Distortion: signal-to-noise ratio

- error (noise) $r(n) = x(n) - y(n)$
- variances $\sigma_x^2, \sigma_y^2, \sigma_r^2$
- power for signal with pdf $p(x)$ and range $-V \ldots + V$:

$$\sigma_x^2 = \int_{-V}^{+V} (x - \bar{x})^2 p(x) dx$$

- $\sigma_u^2 = \frac{1}{M} \sum_{n=1}^{M} u^2(n)$
- SNR = $6.02N - 1.73$ for uniform quantizer with N bits

Distortion: other measures

- SNR *not* a good measure of perceptual quality
- ➠ segmental SNR: time-averaged blocks (say, 16 ms)
- frequency weighting
- subjective measures:
  - A-B preference
  - subjective SNR: comparison with additive noise
  - MOS, DRT, DAM, P.861, . . .

Audio on Solaris

- `cat sample.au > /dev/audio` works; see samples in `/usr/demo/SOUND/sounds/`
- `audiocontrol` to change ports and volume
- `audiotool` for recording and playback
- `audioplay` for playing back sound files

Audio on Solaris

```
#include <sys/audioio.h>
ac = open("/dev/audioctl", O_RDWR);
au = open("/dev/audio",    O_RDWR);
AUDIO_INITINFO(&ai);
ai.record.port = AUDIO_MICROPHONE;
ai.play.port   = AUDIO_SPEAKER;
ioctl(ac, AUDIO_SETINFO, &ai);

bytes = read(au, buffer, bytes);

write(au, buffer, bytes);
```

Careful with opening audio input - keep bit bucket handy!

Audio on Solaris

- read() blocks until audio read
- set device to non-blocking if only currently available audio needed
- write() returns when copied to device, but playout may last longer
- typical loop for world's most expensive microphone amplifier:

```
while (1) {
  b = read(au, buffer, bytes);
  /* audio processing */
  write(au, buffer, b);
}
```

- `ioctl(au, AUDIO_DRAIN, 0);` blocks until audio played out
- use select() to handle both network and audio input

Audio on Linux

- `/dev/audio` for μ-law device
- `/dev/dsp` for general samples
- `aumix` allows to configure mixer
- use `fuser -v /dev/dsp` to find out who's using the device
- watch for endianness - Linux supports both LE and BE
- guide at http://www.4front-tech.com/pguide/audio.html

Audio on Linux: example

```
#include <ioctl.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/soundcard.h>
#define BUF_SIZE    4096
int format = AFMT_S16_LE, stereo = 1, speed = 11025;
if ((audio_fd = open("/dev/dsp", open_mode, 0)) == -1) {
  /* error */
}
if (ioctl(audio_fd, SNDCTL_DSP_SETFMT, &format) == -1) {
}
if (ioctl(audio_fd, SNDCTL_DSP_STEREO, &stereo) == -1 {
}
if (ioctl(audio_fd, SNDCTL_DSP_SPEED, &speed) == -1) {
}
/* set to full duplex */
if (ioctl(audio_fd, SNDCTL_DSP_SETDUPLEX, 0) == -1) {
}
```

Audio on Windows

- uses mixer
- based on callbacks
- callback copies wave blocks
- also: DirectX

Java sound interface

- Java Sound API (java.sun.com/products/java-media/sound/)
- higher layer: Java Media Framework (JMF), includes RTP
- javax.sound.sampled.spi as *service provider*

Java sound API: example

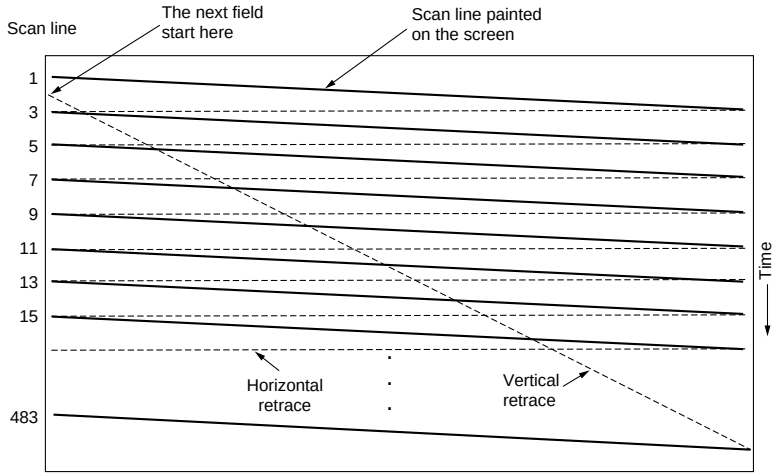
```
TargetDataLine line;
DataLine.Info info = new DataLine.Info(TargetDataLine.class,
  format); // format is an AudioFormat object
byte[] data = new byte[line.getBufferSize()/5];

if (!AudioSystem.isLineSupported(info)) {
  // Handle the error ...
}
// Obtain and open the line.
try {
  line = (TargetDataLine) AudioSystem.getLine(info);
  line.open(format, bufferSize);
} catch (LineUnavailableException ex) {
  // Handle the error ...
}
// Begin audio capture.
line.start();
numBytesRead = line.read(data, offset, data.length);
```

Analog video

- black & white (RS170) + color burst subcarrier
- 15,750 Hz horizontal scanning

system	where	lines (disp.)	size	f/s	bw
NTSC	US, Japan	525 (483)	4:3	30	6 MHz
PAL	Europe	625 (576)	4:3	25	8 MHz



reduce flicker ➠ interlace scanning: 2 fields/frame

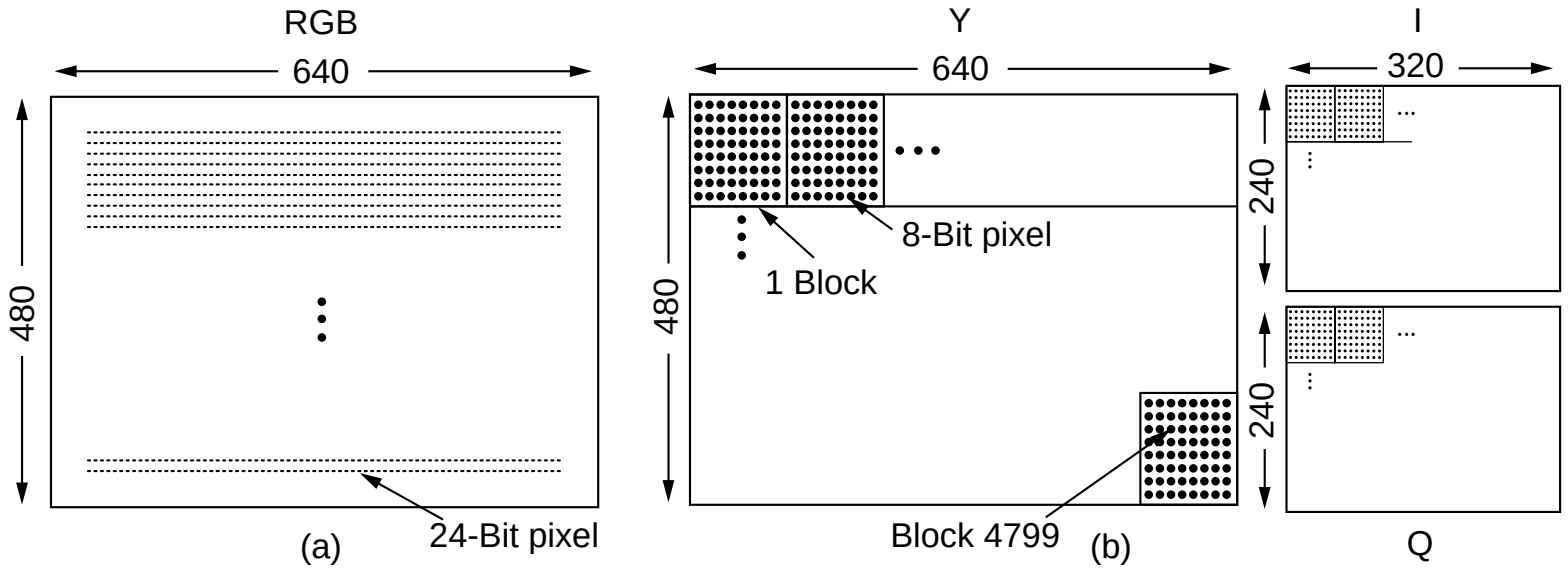
Digital video

format	lines	pixels
SCIF	480	704
CIF	288	352
QCIF	144	176

- motion video: 5 to 30 f/s
- camera ➠ Red, Green, Blue ➠ chrominance, luminance
- YUV: Y = luminance, UV: color difference (red, blue − Y)

Digital video

eye more sensitive to luminance ➠ subsample chrominance: 2:1:1, 4:1:1, 4:2:2



treat chrominance and luminance differently

Video coding

lossless (entropy): (X ray images!) run-length, statistical encoding (Huffman coding, ...)

lossy: ➠ exploit spatial redundancy

transform: $\rightarrow$ frequency domain, higher quantization steps for higher frequencies

vector quantization: map NxN block into N^2-dimensional space and find closest in *codebook*

model-based: geometric description ➠ very low bit-rate

Images

X-ray digitization: 4000 pixels x 4000 lines, at 12 bits/pixel

slide: 120-150 dpi for 6.75” x 10.25”

70 mm movie: 2210 lines

35 mm film: 1753 lines

slide film: 100 lines/mm (2500 lines/inch)

HDTV: 1125 lines

fax: 200 lines/inch

Video coding: JPEG (Joint Photographic Experts Group)

- individual (still) pictures
- lossless or lossy
- typically about 2 Mb/s for video stream
- discrete cosine transform (DCT): samples ➠ blocks (16x16 Y, 8x8 UV) ➠ frequency domain 2D matrix, (0,0) = “DC”

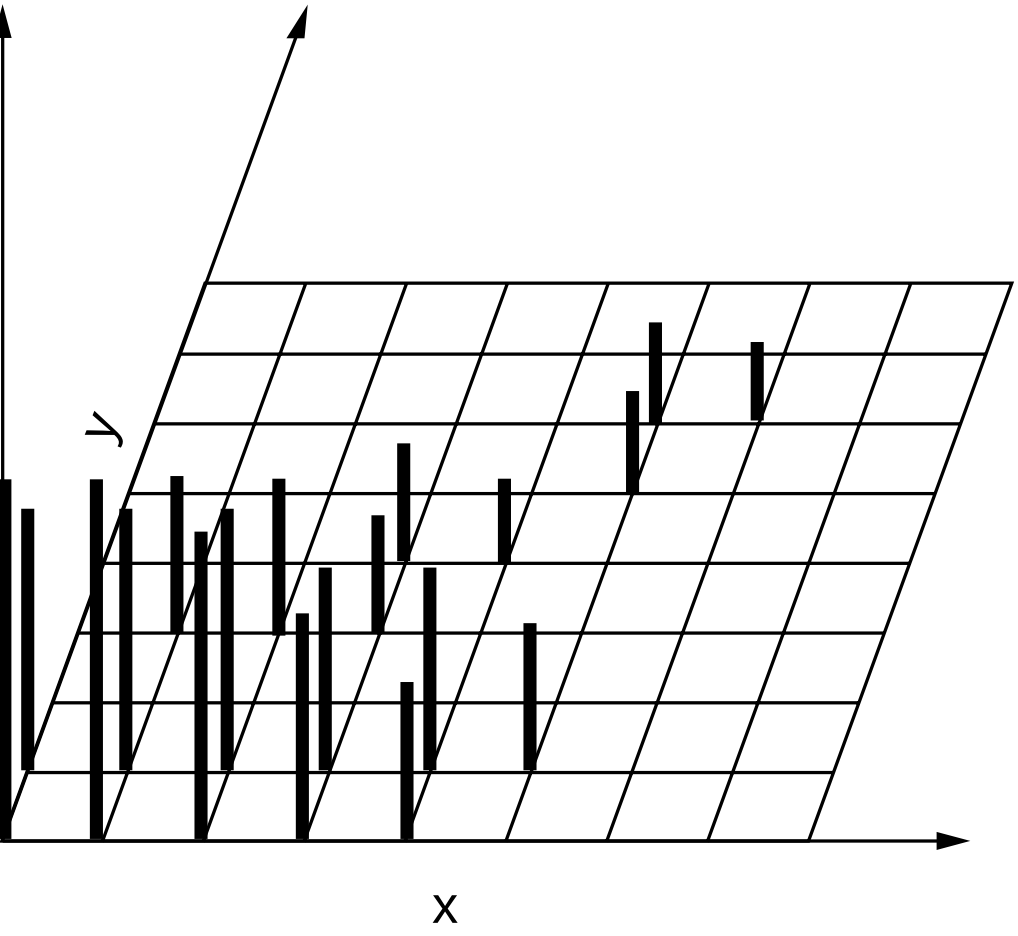
$$B_{u,v}(i,j) = \cos\left(\frac{(2i+1)u\pi}{16}\right)\cos\left(\frac{(2j+1)v\pi}{16}\right)$$

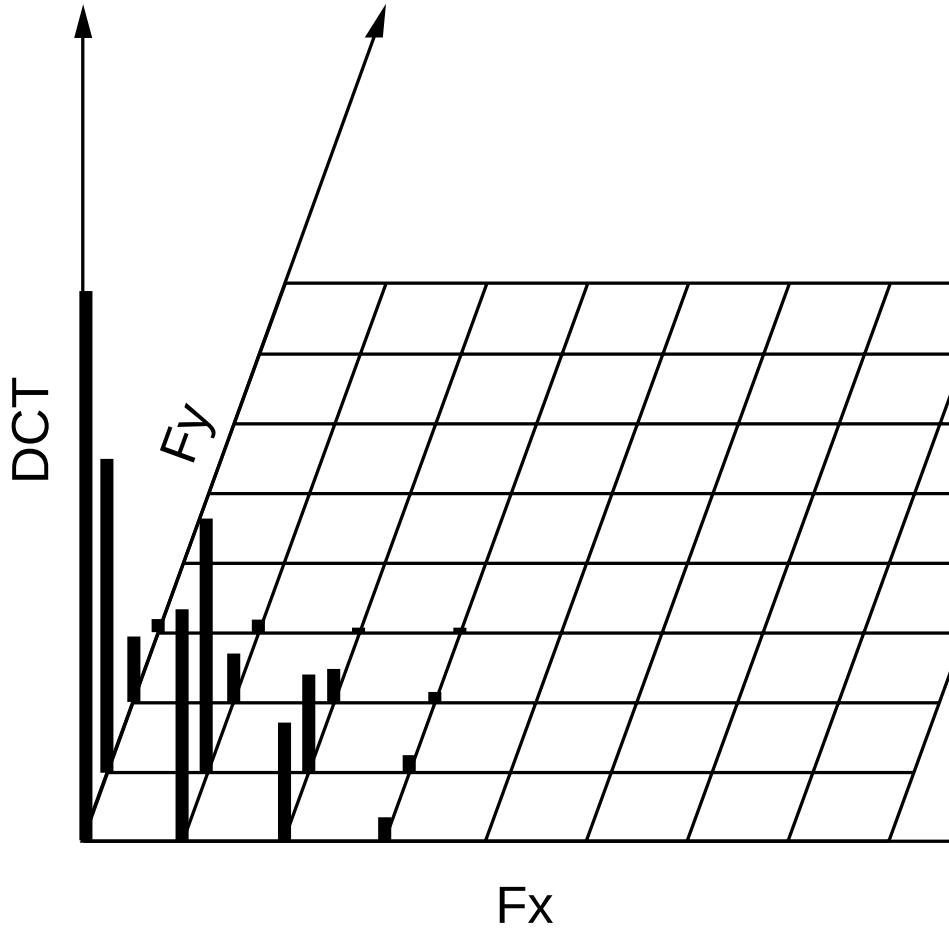
with the transformed image

$$F(u,v) = \frac{2}{N}C(u)C(v)\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)B_{u,v}(i,j)$$

Video coding: JPEG

code frequency sampled with different resolution Q





Video coding: JPEG (Joint Photographic Experts Group)

DC or average (0,0) value: difference to previous block

DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized coefficients

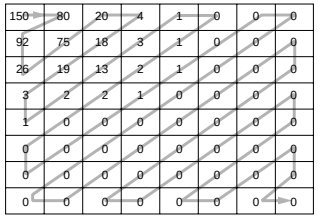
150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

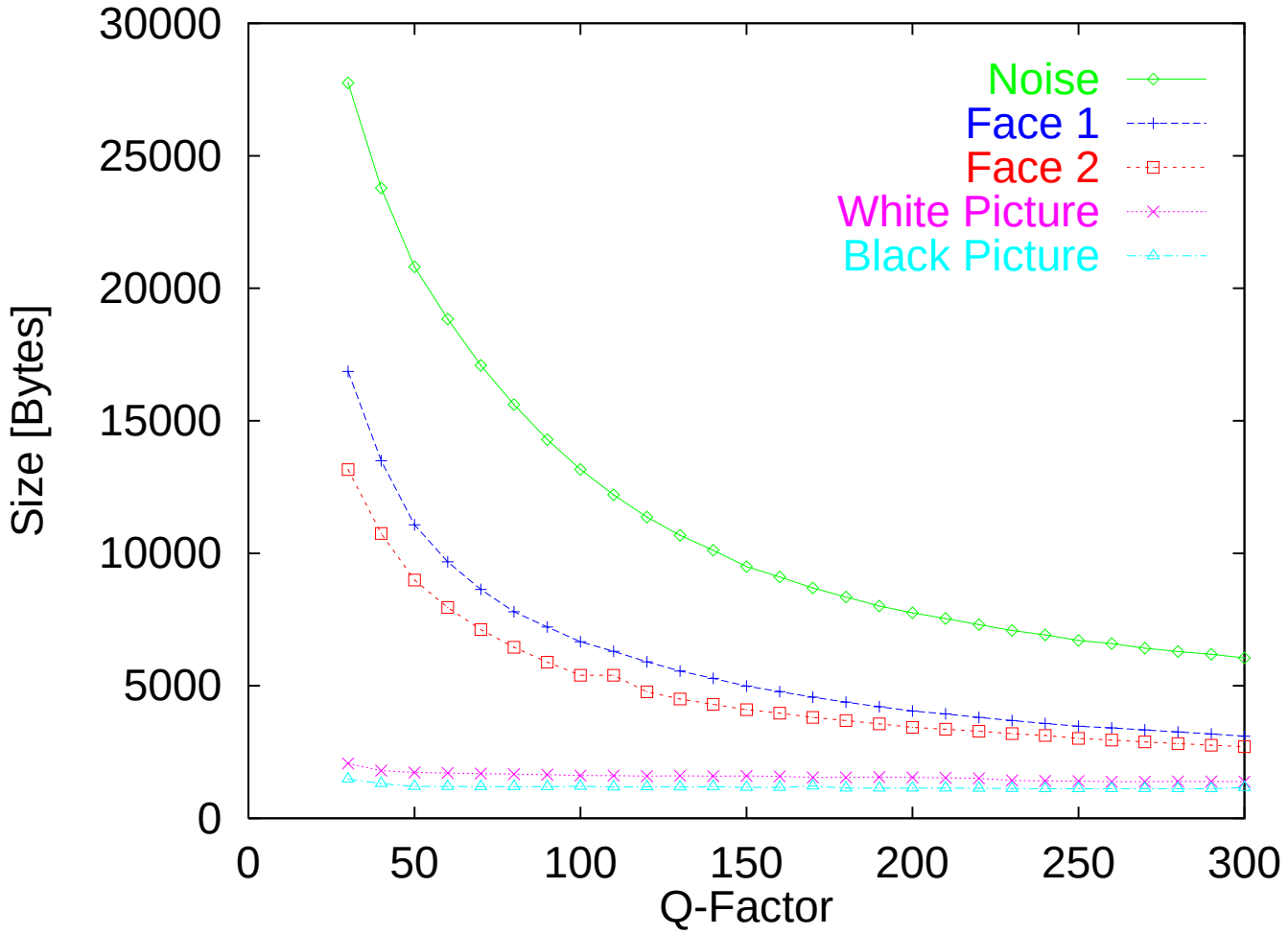
JPEG

- zig-zag scan:



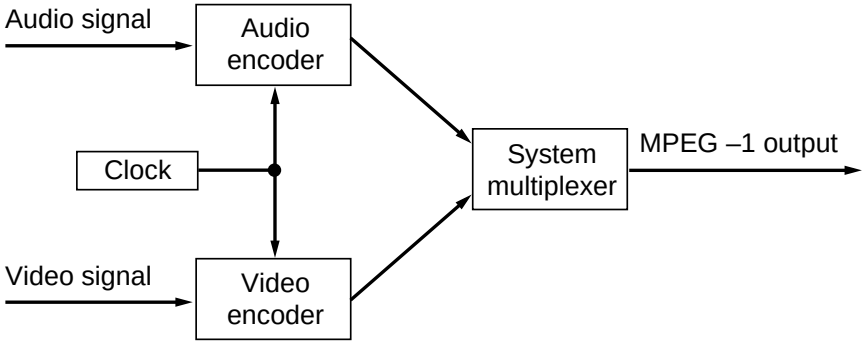
- run-length coding: group all zeros
- Huffmann coding
- See also `http://www.cs.columbia.edu/~hgs/video`

JPEG bandwidth



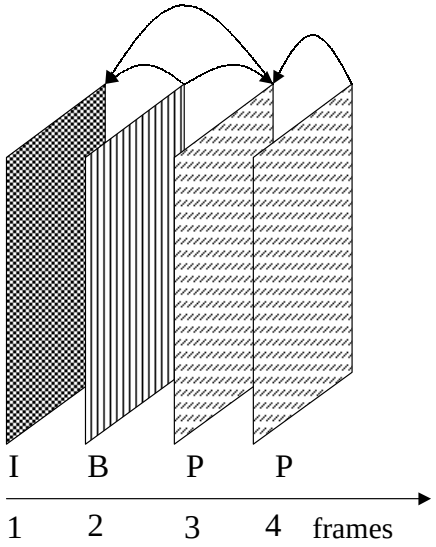
MPEG (Motion Picture Experts Group)

Audio and video:



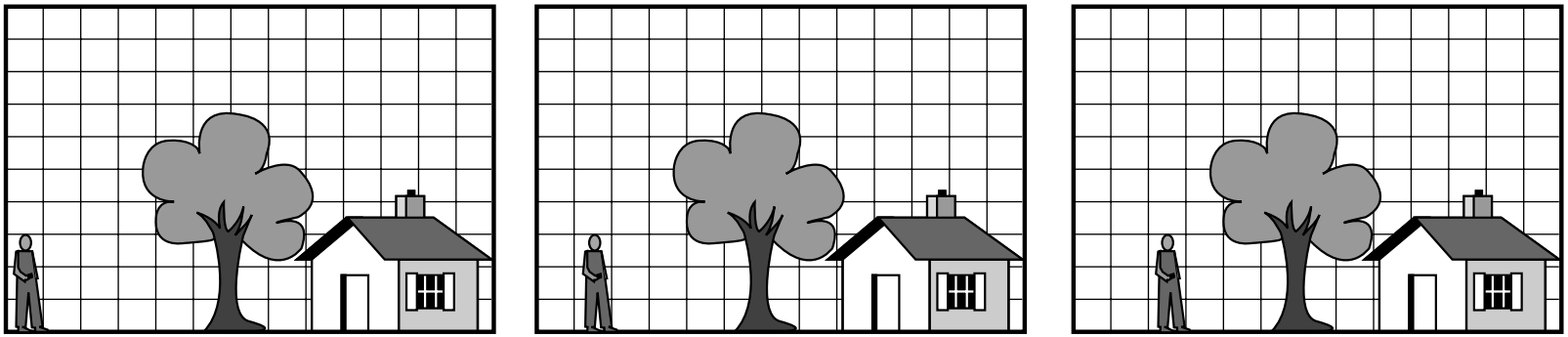
- ➠ JPEG + motion compensation ≈ H.261, MPEG video
- MPEG-1: 1.2 Mb/s fixed rate (CD ROM)
- MPEG-2: higher resolutions (HDTV), scaling
- image prediction: intra (I), forward (P), bidirectional (B) ➠ IBBPBBPBBPBBIBBPBBPB
- need I frames for error resiliency, joining movie in the middle

MPEG I, P, B



B frames need to wait for next frame.

MPEG: motion compensation



- transmit “motion vectors” to account for panning and zooming ➠ hard to find (must try lots), easy to decode

H.261 video codec

- ISDN ($n\times$ 64 kb/s) conferencing $\longrightarrow$ lower delay
- conditional replenishment: only transmit blocks that are different
- motion vectors for each 16x16 macroblock: ± 15 integer pixels
- GOB: 11 macroblocks H, 3 macroblocks V, marked by *start code*

H.263 video codec

- sub-QCIF (128 $\times$ 96), 4CIF (704x576), 16CIF (1408x1152)
- motion prediction outside frame
- advanced prediction mode: 4 vectors for each 8x8 block
- advanced intra prediction – within picture
- slice-structured mode: non-overlapping rectangles
- scalability: temporal (B frames), SNR, spatial

HDTV

- subset of MPEG-2 video compression, Dolby AC-3 audio compression
- vestigial sideband modulation (8-VSB) of 19 Mb/s or 16-VSB for two channels in CATV
- formats:

1280 x 720	24, 30, 60 Hz progressive scan
1920 x 1080	24, 30 progressive, 60 Hz interlaced

- MPEG-2 transport stream: fixed-length 188-byte packets (4x47 ATM cells)
- one channel = one or more programs

Multiplexing

Pack multiple streams into a single lower layer

- IETF: MIME, RTP (later)
- ITU: H.221 (synchronous, 80x8)
- MPEG: elementary, transport, program
- file formats: AVI, QuickTime

Characteristics of digital audio and video

	audio	video
rate	5.3...64...1500 kb/s	0.2...1.5 ...19 Mb/s
loss tolerance	$\leq 5\%$	10^{-5}...10%
packet size	small	large
traffic	constant + silences	variable bit rate

Audio traffic models

talkspurt: constant bit rate: one packet every 20...100 ms ➠ mean: 1.67 s

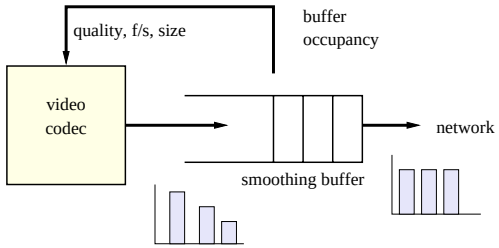
silence period: usually none (maybe transmit background noise value) ➠ 1.34 s

➠ for telephone conversation, both roughly exponentially distributed

- double talk for “hand-off”
- may vary between conversations... ➠ only in aggregate

Video traffic models

- easy case: fit into constant bit rate



- alternative: variable rate ➠ mux gain of $\approx 2 \ldots 6$
- short time scales: packets within slice or frame
- medium time scales: I, B, P packet pattern
- longer time scales: scene changes (every few seconds) ➠ higher rate
- looks similar at all time scales, long-term correlation, *heavy-tailed distribution* ➠ *self-similar*
- but: for short queues, long-term correlations don't matter