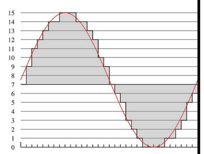
ESE1500 Spring 2023

Penn Engineering

ESE



Lecture #12 – Psychoacoustic Compression Algorithm

ESE 1500 – Digital Audio Basics

Based on slides © 2009—2023 DeHon, Koditschek
Additional Material © 2014 Farmer

1

1

ESE1500 Spring 2023

Lecture Topics

- **Review**
- **Exploiting psychoacoustics in MP3**
  - Review Tricks
  - Formulate Optimization
  - **Algorithm for Adaptation**
    - **Basic (part 2)**
    - **Refinements (part 3)**
- **References**

2

2

ESE1500 Spring 2023

Review and Preclass Setup/Reminder

3

3

ESE1500 Spring 2023

Knobs we can turn

- **Amplitude quantization**
  - Per band
- **Frequency quantization**
  - Per band
- **Frequencies kept**
  - Per band
- **…and can perform lossless compression**

4

4

ESE1500 Spring 2023

MP3 Encoding Process

- **All MP3 files broken into "Frames"**
  - Each frame stores 1152 Audio Samples
  - Lasts for 26 ms
  - Frame also divided further into 2 "granuels"
    - Each granuel contains 576 samples

5

5

ESE1500 Spring 2023

PCM → MP3

- 44,300 samples/s
- 16b
- 26ms window (frame)
- → 18,429 bits/frame
- MP3: 128Kb/s stereo → 64Kb/s per audio channel
- → 1704 bits/frame

6

6

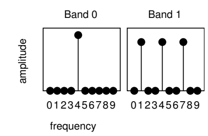
Optimization Problem

- **How fit in the resource constraints (128Kb/s) while maximizing goodness (sound quality)?**
- **Quantify bits used:** $\sum_{bands} \sum_{f \in freqs} Bits(f)$
  - Cannot exceed 128Kb/s
  - = 1,704 b / 26ms frame / channel
- **Quantify goodness: minimize**
  $$\sum_{f \in freqs} Error(f) \times W(f)$$

7

Seen

- **May want to do something smarter than**
  - Allocating fixed number of frequencies per band
  - Allocating fixed quantization to a band
- **Like to adapt our encoding to the data**
  - If more Huffman compressible, we get more frequencies
  - If fewer frequencies suffice for one band,
    - Allow more frequencies for another
    - …or allocate less quantization



8

Optimization Problem

- **How fit in the resource constraints (128Kb/s) while maximizing goodness (sound quality)?**
- **Optimization problems central to engineering**

9

Optimization Problem

- **How fit in the resource constraints (128Kb/s) while maximizing goodness (sound quality)?**
- **Quantify bits used:** $\sum_{bands} \sum_{f \in freqs} Bits(f)$
  - Cannot exceed 128Kb/s
  - = 1,704 b / 26ms frame / channel
- **Quantify goodness: minimize**
  $$\sum_{f \in freqs} Error(f) \times W(f)$$

10

Goodness/Sound Quality

- **Error(Amp) = |Orig Amplitude – Encoded|**
  - Whole OrigAmplitude if dropped
  - |Orig Amplitude-Quantize(OrigAmplitude,bits)| if quantized
- **W(freq)**
  - 0 if below hearing threshold
  - 0 if masked
  - Value between 0 and 5 if partially masked in critical band
  - Really depend on what already encoded

$$\sum_{f \in freqs} Error(f) \times W(f)$$

11

Example Weight Function W(f)

- **W(f)=CBWeight*Mask**
- **Mask = 0 if MaxAmp-FreqAmp>3**
  - **1 otherwise**

12

Penn Engineering

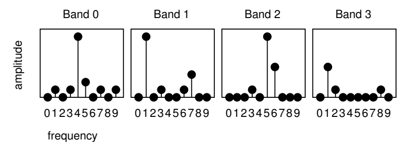
ESE

Part 2

Basic Incremental Encoding Algorithm

Example

- Simplified example with 4 critical bands

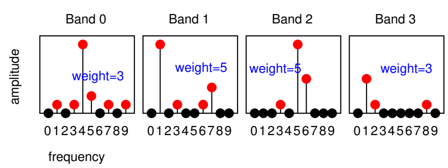


Approach

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

$$\sum_{f\in freqs} Error(f)\times W(f)$$

- Start with nothing
- Start with bitbudget
- Incremental moves to improve



Moves

- Select a frequency to encode
  - Look at first
- Assign more bits to a band for quantization
  - Add in second

Greedy Incremental Frequency Selection

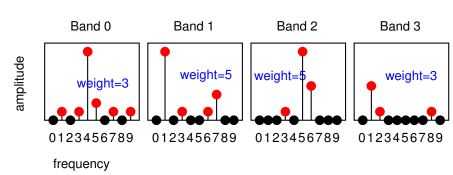
- Start with nothing encoded
- Start with bits=0
- While (bits<bitbudget)
  - Identify frequency with maximum current error
  - Identify bits required to encode frequency (encode_bits)
  - If (bits+encode_bits<bitbudget)
    - Add frequency to encoding
    - Update bits (bits+=encode_bits)
  - Else break

Approach

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

$$\sum_{f\in freqs} Error(f)\times W(f)$$

- Start with nothing
- Start with bitbudget

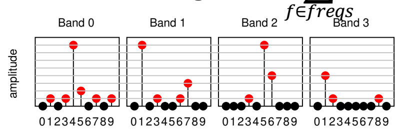


APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget

$$\sum_{f\in freqs} Error(f)\times W(f)$$



Bits = 0
Error = 190

	Band 0	Band 1	Band 2	Band 3
	11 8 2 11	8 1 1 3	1 8 4	41 1
sum	14	13	13	6
weight	3	5	5	3
weighted	42	65	65	18

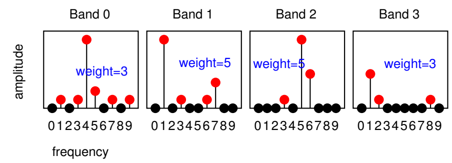
19

APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget
- While(bits<bitbudget)
  - Identify Largest Error reduction component: Error(freq)×W(freq)

$$\sum_{f\in freqs} Error(f)\times W(f)$$



Bits = 0
Error = 190

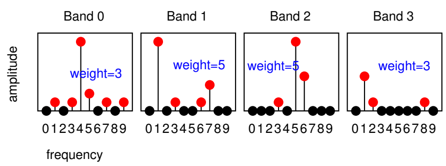
20

APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget
- While(bits<bitbudget)
  - Identify Largest Error component: Error(freq)×W(freq)
  - What do to reduce error?

$$\sum_{f\in freqs} Error(f)\times W(f)$$



Bits = 0
Error = 190

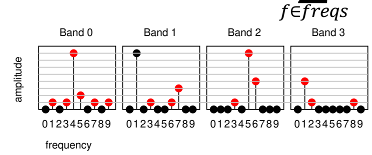
21

APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget

$$\sum_{f\in freqs} Error(f)\times W(f)$$



Bits = 3+6=9
Error = 125

	Band 0	Band 1	Band 2	Band 3
	11 8 2 11	0 0 0 0	1 8 4	41 1
sum	14		13	6
weight	3	5	5	3
weighted	42	0	65	18

22

APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget
- While(bits<bitbudget)
  - Identify Largest Error component: Error(freq)×W(freq)
  - What do to reduce error?

$$\sum_{f\in freqs} Error(f)\times W(f)$$

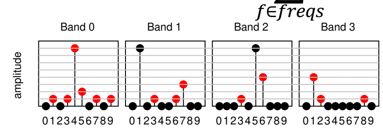
23

APPROACH

$$\sum_{bands}\sum_{f\in freqs} Bits(f)$$

- Start with nothing
- Start with bitbudget

$$\sum_{f\in freqs} Error(f)\times W(f)$$



Bits = 18
Error = 60

	Band 0	Band 1	Band 2	Band 3
	11 8 2 11	0 0 0 0	0 00	41 1
sum	14			6
weight	3	5	5	3
weighted	42	0	0	18

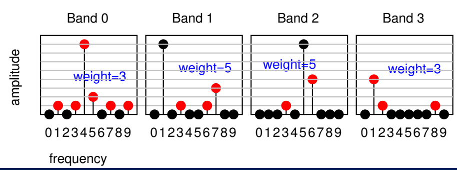
24

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

$$\sum_{f \in freqs} Error(f) \times W(f)$$

- Start with nothing
- Start with bitbudget
- While(bits<bitbudget)
  - Identify Largest Error component: Error(freq)×W(freq)
  - What do to reduce error?



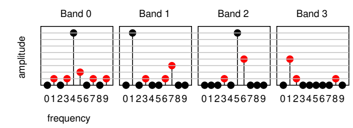
25

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

$$\sum_{f \in freqs} Error(f) \times W(f)$$

- Start with nothing
- Start with bitbudget
- (still using 3b/band)



Bits = 27
Error = 18

	Band 0	Band 1	Band 2	Band 3
		0 0 0 0	0 00	41 1
sum	0			6
weight	3	5	5	3
weighted	0	0	0	18

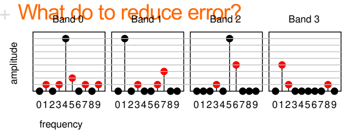
26

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

$$\sum_{f \in freqs} Error(f) \times W(f)$$

- Start with nothing
- Start with bitbudget
- While(bits<bitbudget)
  - Identify Largest Error component: Error(freq)×W(freq)
  - What do to reduce error?



Bits = 27
Error = 18

	Band 0	Band 1	Band 2	Band 3
		0 0 0 0	0 00	41 1
sum	0			6
weight	3	5	5	3
weighted	0	0	0	18

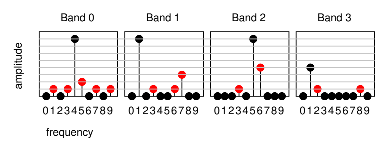
27

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

(still with 3b in all bands)

$$\sum_{f \in freqs} Error(f) \times W(f)$$



Bits = 36
Error = 0

	Band 0	Band 1	Band 2	Band 3
		0 0 0 0	0 00	00 0
sum	0			0
weight	3	5	5	3
weighted	0	0	0	0

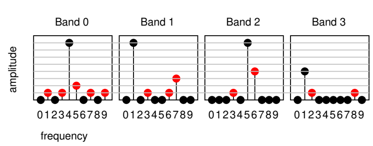
28

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

With 2b in bands 0 and 3?

$$\sum_{f \in freqs} Error(f) \times W(f)$$



Bits = ?
Error = 0

	Band 0	Band 1	Band 2	Band 3
		0 0 0 0	0 00	00 0
sum	0			0
weight	3	5	5	3
weighted	0	0	0	0

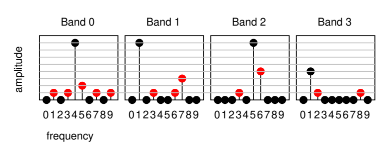
29

Approach

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

With 2b in bands 0 and 3:
(lucky here 2b captures maskees)

$$\sum_{f \in freqs} Error(f) \times W(f)$$



Bits = 34
Error = 0

	Band 0	Band 1	Band 2	Band 3
		0 0 0 0	0 00	00 0
sum	0			0
weight	3	5	5	3
weighted	0	0	0	0

30

Penn Engineering

ESE

Part 3

Refine Encoding Algorithms

31

Also Quantization

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

$$\sum_{f \in freqs} Error(f) \times W(f)$$

- **To keep simple, assumed fixed quant.**
- **Incrementally assign bits**
- **While(bits<bitbudget)**
  - Identify Largest Error component: Error(freq)×W(freq)
  - Assign more bits to that frequency
    - Go from 0 bits to 1 bit
      - 1 bit to 2 bits

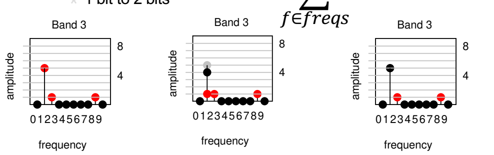
32

Also Quantization

$$\sum_{bands}\sum_{f \in freqs} Bits(f)$$

- **While(bits<bitbudget)**
  - Identify Largest Error component: Error(freq)×W(freq)
  - Assign more bits to that frequency
    - Go from 0 bits to 1 bit
      - 1 bit to 2 bits

$$\sum_{f \in freqs} Error(f) \times W(f)$$



0b -- error=7

1b quant -- error=1 (1+6) encode

3b quant -- error=0 (3+9) encode

33

Greedy Incremental Frequency Selection

- **Start with nothing encoded**
- **Start with bits=0**
- **While (bits<bitbudget)**
  - Identify frequency with maximum current error
  - Identify bits required to encode frequency (encode_bits)
  - If (bits+encode_bits<bitbudget)
    - Add frequency to encoding
    - Update bits (bits+=encode_bits)
  - Else break

34

Greedy Incremental Frequency Selection (Recover Huffman)

- **Start with nothing encoded**
- **Start with bits=0**
- **While (bits<bitbudget)**
  - Identify frequency with maximum current error
  - Identify bits required to encode frequency (encode_bits)
  - If (bits+encode_bits<bitbudget)
    - Add frequency to encoding
    - Re-Huffman encode and update bitbudget
  - Else break

35

Greedy Incremental Frequency Selection (Recover Huffman)

- **Start with nothing encoded**
- **Start with bits=0**
- **While (bits<bitbudget)**
  - Identify frequency with maximum current error
  - Allocate some bits to reduce error
    - Add frequency or Add quantization bits to band
    - Pick one to most reduce the error
    - Re-Huffman encode and update bitbudget
  - If (Huffman_encoded_bits<bitbudget)
    - Add frequency to encoding
  - Else break

36

Adaptive Refinement

- **Rediscovering where to allocate everything every time may be laborious**
  - Often same frequencies persist for more than 26ms
- **Maybe we can get close and adjust?**
  - Use critical band allocation from previous frames as a starting point guess
    - bits, frequencies, quantization
  - Try initial encoding with that

37

Approach (Adaptive)

- **Start with budget guess**
  - Quantization in bands
  - Frequencies to keep in each band
- **Encode, compress**

- **What can we do if takes up < 1704 bits?**
- **What can we do if takes up > 1704 bits?**

38

Finish with Adaptive Fine Tuning

- **<previous slide>**
- **FrameBits=Huffman(FrameFreq)**
- **While ((FrameBits>1704) | (FrameBits<(1704-26))**
  - If (FrameBits>1704)
    - NumFreqs—
    - FrameFreq←RemoveLeastImportant(FrameFreq)
  - Else if (FrameBits<(1704-26))
    - NumFreqs++
    - FrameFreq←AddNextImportant(FrameFreq)
  - FrameBits=Huffman(FrameFreq)

39

Finish with Adaptive Fine Tuning

Could add/subtract more than one freq at a time.
Could keep track of high and low encodings.

- **<previous slide>**
- **FrameBits=Huffman(FrameFreq)**
- **While ((FrameBits>1704) | (FrameBits<(1704-26))**
  - If (FrameBits>1704)
    - NumFreqs—
    - FrameFreq←RemoveLeastImportant(FrameFreq)
  - Else if (FrameBits<(1704-26))
    - NumFreqs++
    - FrameFreq←AddNextImportant(FrameFreq)
  - FrameBits=Huffman(FrameFreq)

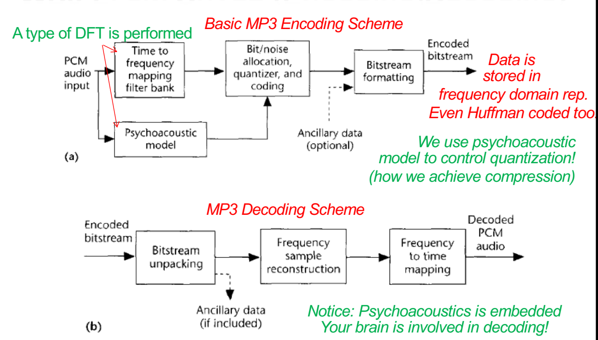
40

Roundup

Perceptual Coding & MP3

41

MP3 Flow Chart (Encoding/Decoding)



42

Compare to Lab

- **Lecture 11 & 12 (last time, this time)**
  - Help understand more what real MP3 encoding looks like
  - Thinking about fixed rate
    - And adaptation for variable rate encoding from Huffman
  - Formulating masking explicitly
    - But simplistic
  - Illustrating Optimization Approaches
- **Lab 6**
  - Capture spirit of reducing frequencies
  - Simplified – only taking loudest fraction in each band
    - Rather than being rigorous about masking
    - Or trying to hit some fixed rate

Big Ideas

- **Can use pyschoacoustics to compress audio**
  - Eliminate portions of signal that human's don't notice
- **Optimization**
  - Identify Design Space (knobs)
  - Identify Costs and Constraints
  - Formulate quantitatively
  - Algorithms to approach
  - Iterative/adaptive approach
    - Deal with effects that aren't completely predictable

Learn More

- **Optimization –**
  - continuous mathematical optimization ESE2040, ESE5040, ESE6050
  - discrete optimization CIS1210, CIS3200
- **Signal processing – ESE2240**

Coming Up

- **Feedback**
  - including first part of lab 6 from Monday before break
- **Lab 6 continue today**
  - 2 week lab
  - Brings together first half of course
  - Formal report due next Monday
- **On to logic on Wednesday**

References

Tutorials on Psychoacoustic Coding (in increasing order of abstraction and generality)

D. Pan, M. Inc, and I. L. Schaumburg. A tutorial on MPEG/audio compression. IEEE multimedia, 2(2):60–74, 1995.

Nikil Jayant, James Johnston, and Robert Safranek. Signal compression based on models of human perception. Proceedings of the IEEE, 81(10):1385–1422, 1993.

V. K. Goyal. Theoretical foundations of transform coding. IEEE Signal Processing Magazine, 18(5):9–21, 2001.

Lightweight Overview of MP3

Rassol Raissi. The theory behind mp3. Technical report, MP3' Tech, December 2002.

Scientific Basis of MP3 Coding Standard

J. D. Johnston. Transform coding of audio signals using perceptual noise criteria. IEEE Journal on selected areas in communications, 6(2):314–323, 1988.