

Sound Compression and the Discrete Cosine Transform

Sarah Edge Mann
Program in Applied Mathematics
University of Arizona

Introduction

In this age of digital media, we have the ability to digitally capture sound and images at astonishingly high fidelity. However, this media can require such vast amounts of storage space that it is prohibitive for wide spread use. For example, the raw digital recording of a three minute song might have a file size of 29.5 MB. A 2GB Apple iPod Shuffle could hold a mere 68 songs of this size, whereas Apple claims it can hold 500 songs. The reason for this discrepancy is that Apple assumes that the sound files put on the iPod will be in a compressed format and will thus require only about 4 MB of space per song. Similarly, 15 terabytes are required to capture a mere 15 minutes of raw digital video. This is the equivalent of 250 DVDs worth of data, and yet, in practice, we are able to put an entire 2 hour movie on a single DVD. Again, this feat is achieved through data compression.

There are a wide array of algorithms that will take a raw digital sound file and write it in a different format using less space (the MP3 and AAC are two such examples). There are two categories of algorithms that compress sound (or any other) files. The first is called lossless compression: the compression algorithm attempts to represent the file in a more efficient way without losing any information. The second type is called lossy compression: the compression algorithm creates a smaller file, but in so doing loses some information.

Goal

Develop a lossy compression algorithm for sound data based on the discrete cosine transform, quantization, k -means clustering, and arithmetic encoding that minimizes the size of the compressed file while maintaining the highest possible sound quality.

Digital Sound Recording

- Physically, sound is a standing wave of pressure propagating through a medium, typically air.
- When sound is recorded digitally, the amplitude of the pressure inside a microphone is measured many times per second (typically around 44,100 Hz).
- The precision d of the recording is the number of bits used to represent the amplitude as a fixed point number (typically 8 or 16-bit).
- The file size is the number of bytes required to store the digital recording

$$\text{file size} = \frac{f_s \times t \times d}{8} \text{ bytes}$$

where f_s is the sampling frequency, and t is the length of the recording in seconds. (Note: 1 byte = 8 bits)

Measuring Error

Since we are creating a lossy compression algorithm, we expect to introduce some error in the process of compressing then decompressing the sound data. It is important to establish a measure with which we may quantify the error introduced, then seek to minimize this error. We chose to use the **signal to noise ratio** (SNR) defined by

$$\text{SNR} = 20 \cdot \log_{10} \frac{\| \text{signal} \|_2}{\| \text{error} \|_2}$$

where the signal is obtained by first compressing the original sound data, then decompressing it and the error = signal - original data. A larger SNR corresponds to better sound quality.

The Discrete Cosine Transform of Sound Data

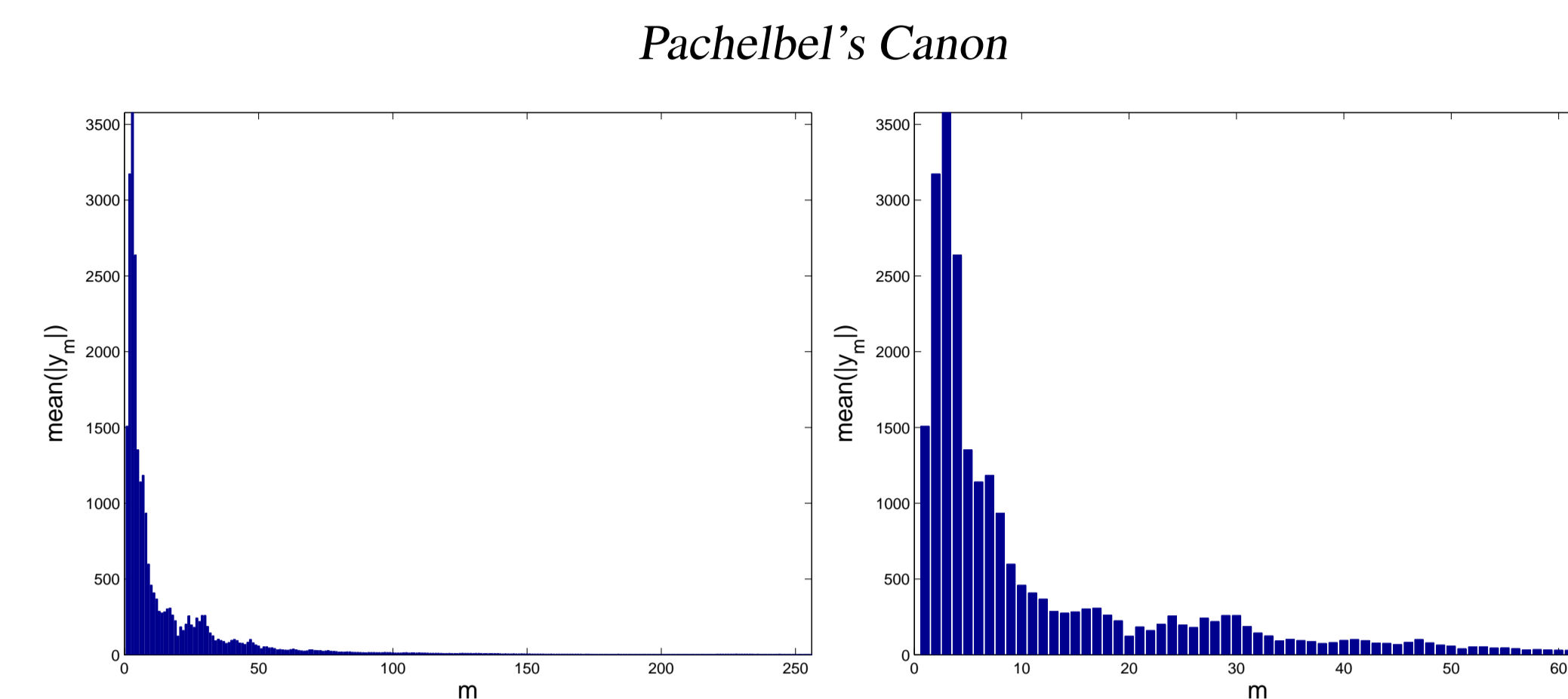


Figure 1: These figures show a profile of the DCT Coefficients of an example sound files, 20 seconds of Pachelbel's Canon. The sample was broken into "packets" of 256 data points, approximately 5 milliseconds of sound, and the DCT of each packets was performed separately. The figure shows the mean value across all packets of the absolute value of each DCT coefficient as a function of coefficient number m . Lower coefficient numbers correspond to lower frequency cosine components. The left hand figure shows the means for all 256 DCT coefficients, the right hand figure shows the means for the first 64.

Observation: Low frequency DCT coefficients are of vastly higher amplitudes than high frequency coefficients. Most of the energy of the sound data is contained in the first few DCT coefficients.

A First Attempt at Compression

1. Break sound data into packets of 256 data points, take DCT of each packet.
2. Round all DCT coefficients to integers.
 - Makes algorithm 'lossy' but increases compression.
3. Arithmetically encode and transmit only first portion of DCT coefficients for each packet.

Fraction Transmitted	Pachelbel's Canon		Interview	
	SNR	Compression	SNR	Compression
1	65.8	43.7 %	76.9	49.9 %
3/4	46.8	35.7 %	60.9	42.4 %
1/2	41.4	26.7 %	40.7	31.7 %
1/4	32.2	15.6 %	22.9	17.4 %

Table 1: Compression rate and SNR achieved by this compression algorithm on two sample files.

Compression with Quantization

1. Break sound data into packets of 256 data points, take DCT of each packet.
2. Divide all DCT coefficients by some constant ϵ and round them to integers.
 - This process is called **quantization**.
 - Makes algorithm 'lossy' but increases compression.
3. Arithmetically encode and transmit signal.

ϵ	Pachelbel's Canon		Interview	
	SNR	Compression	SNR	Compression
1	65.8	43.7 %	76.9	49.9 %
10	45.8	23.5 %	57.7	30.1 %
18	41.2	18.4 %	53.1	25.7 %
25	39.0	15.9 %	50.5	23.3 %

Table 2: Compression rate and SNR achieved by quantizing the DCT coefficients using some fixed ϵ . Note: $\epsilon = 1$ is equivalent to keeping all DCT coefficients in the previous algorithm.

Compression with Clustering of DCT Coefficients

Observation We can achieve further gains in compression by identifying a few groups of similar looking data, and arithmetically encoding each of these groups separately.

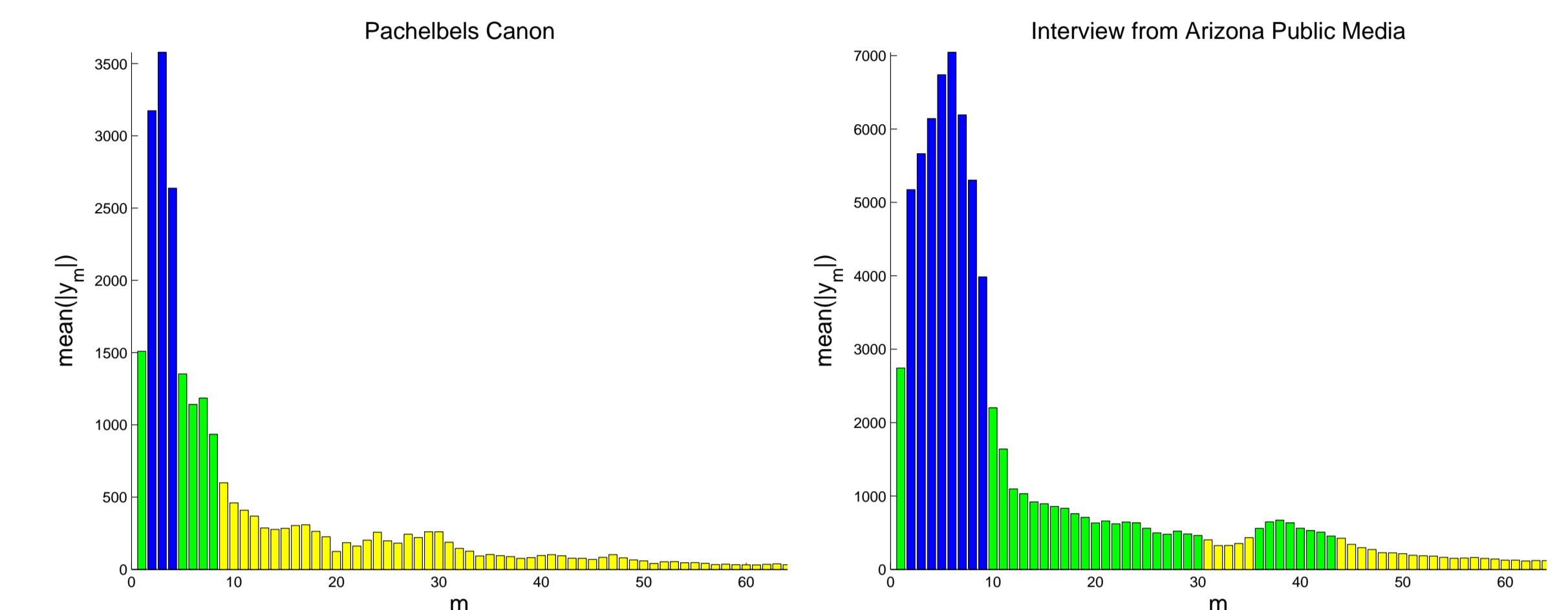
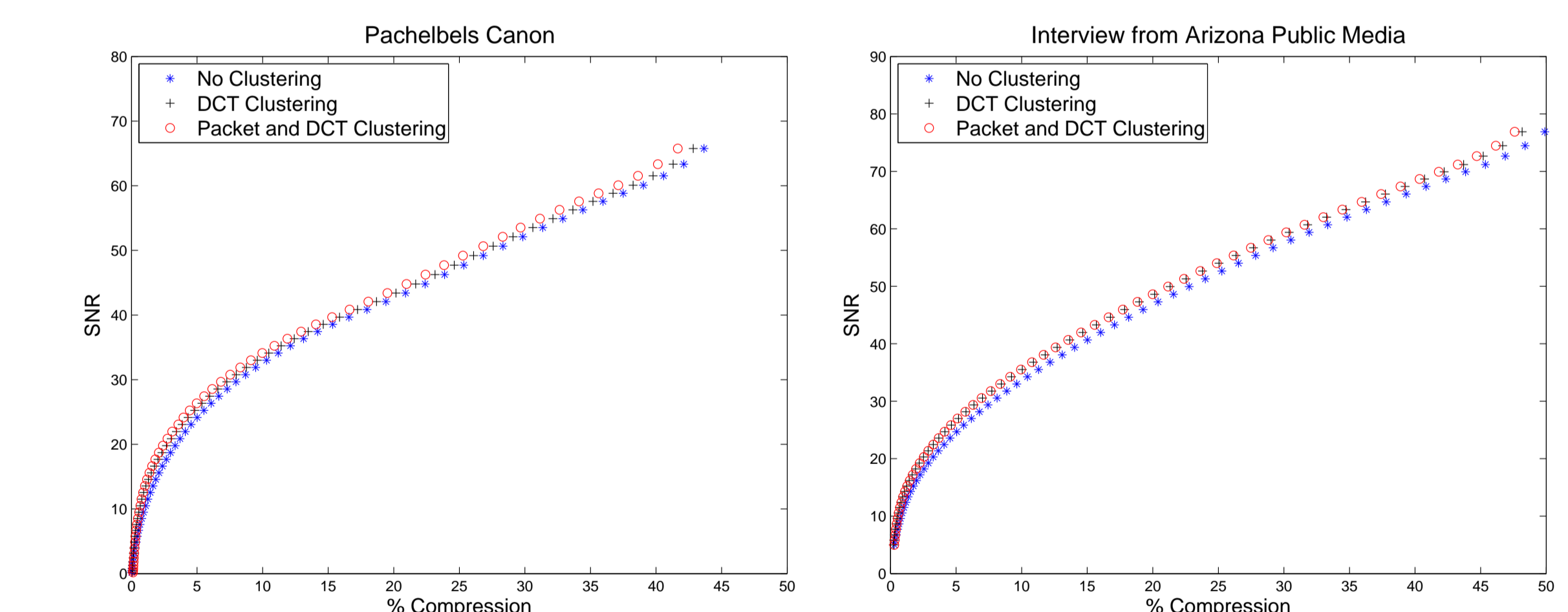


Figure 2: Clustering of the DCT coefficients. Note that only the first 64 coefficients are shown; the remaining coefficients are assigned to the yellow cluster.

Algorithm

1. Break sound data into packets of 256 data points, take DCT of each packet.
2. Divide all DCT coefficients by some constant ϵ and round them to integers.
3. Cluster DCT coefficients into three groups using one dimensional k -means clustering.
4. Arithmetically encode and transmit each cluster.

Comparison of Algorithms



- Clustering improves the performance of the algorithm over quantization with no clustering.
- These algorithms are competitive with the MP3 compression algorithm.
 - MP3 compresses these samples to $\sim 18\%$ of original size with an SNR of ~ 26 .
 - Our algorithm can compress these samples to $\sim 6\%$ of original size with an SNR of ~ 26 .
 - Our algorithm can compress these samples to $\sim 18\%$ of original size with an SNR or ~ 44 .

Acknowledgements

This project was done in collaboration with Priya Prasad. We would like to thank Dr. Marek Rychlik for advising us. Thanks as well to Anna Latta and Arizona Public Media for providing sample sound files of interviews which we used to test these algorithms.