Lab 6 & 7

Elena Montalvo
Data collected with Darren Ray
ECE3101L - Andrew Pagnon
October 13, 2022

Objective

- Analyze the process of converting a continuous analog signal of an image and the sound to a discrete sample signal
- Be able to explain signal analog-to-digital (ADC) and digital-to-analog (DAC) conversion by using quantization.
- Analyze the process of quantizing a continuous analog or high granularity discrete amplitude of a signal to lower granularity discrete samples
- Compare different quantizing algorithms implemented in Matlab
- Change the quantization of an audio and video signal in MATLAB

Introduction

We use quantization best to progress the signal or quantity by rounding the analog signal to be processed by the computer. We could think about quantization as an equation that has different values for the area or time.

We could think about how we tend to round values in a calculator or weight balance values in a balance. We have to also consider the issue of how things that have more precision tend to have a longer processing time and tend to use more space. Quantization could be also used for compression by shirking the number of pixels or bandwidth This is really important because the higher resolution something is the more storage in memory it might use or the longer it might take to be transmitted. However, it could affect the quality of the things and distort the quality of the things that are being transmitted or processed.

As engineers when it comes to quantizing signals we have to consider sending quality signals while at the same time not taking too much memory in the devices or bandwidth so it

In the Matlab analysis, we also consider the quantizing error that the quantization would have because is loses depending on how many bits are being processed it would lose a certain amount of quality

One way to measure the quality of the Power Signal-to-Noise Ratio (PSNR).

$$PSNR = \frac{PY}{PE}$$

We consider that distortion is

$$Distortion = \frac{1}{PSNR}$$

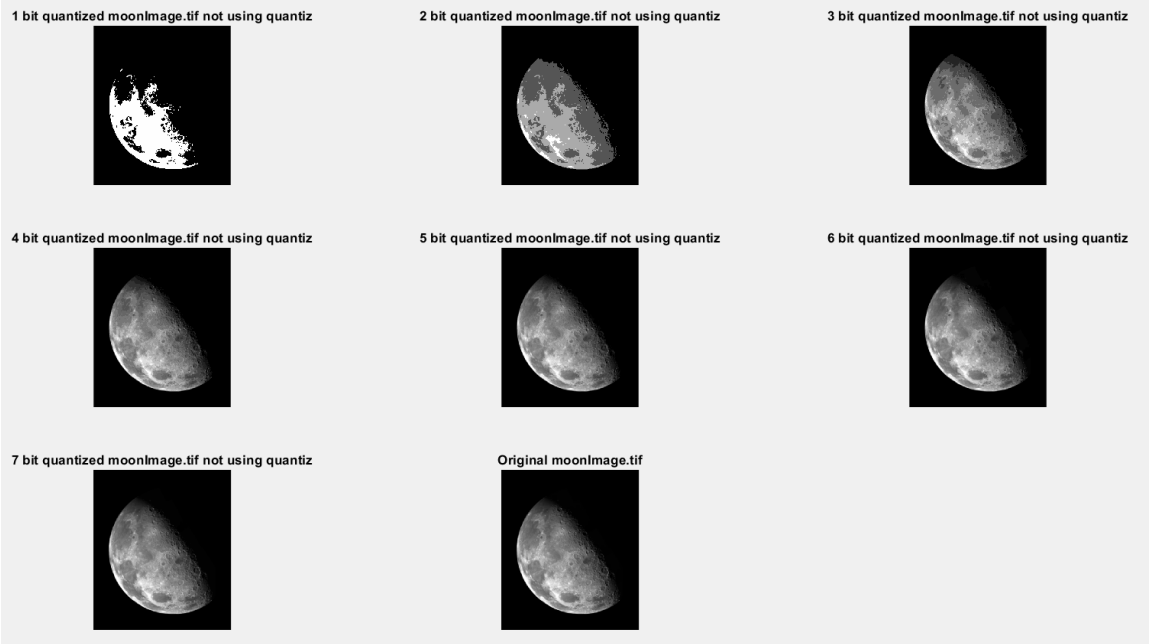
Quantization

Images

For images, we only quantize a black and white images image because the pixels could be references from [000] to [111] to signify the black and white and a possible range of gray tones in between if we were to do closer we would have a more complex range of values and arrays

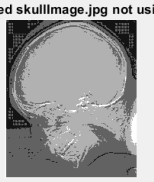
For the images, we could tell how when we used more bits to quantize the images tended to be clear

Quantizing images



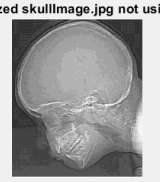
1 bit quantized skullImage.jpg not using quantiz

2 bit quantized skullImage.jpg not using quantiz



3 bit quantized skullImage.jpg not using quantiz

4 bit quantized skullImage.jpg not using quantiz



5 bit quantized skullImage.jpg not using quantiz

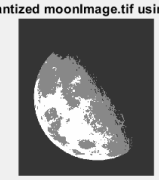
6 bit quantized skullImage.jpg not using quantiz

7 bit quantized skullImage.jpg not using quantiz

Original skullImage.jpg

Imquantize images

2 levels quantized moonImage.tif using imquantize



4 levels quantized moonImage.tif using imquantize

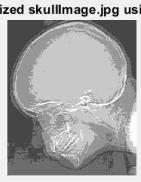
8 levels quantized moonImage.tif using imquantize

16 levels quantized moonImage.tif using imquantize

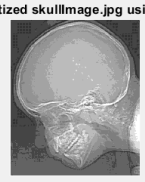
Original moonImage.tif

2 levels quantized skullImage.jpg using imquantize

4 levels quantized skullImage.jpg using imquantize

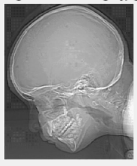


8 levels quantized skullImage.jpg using imquantize



16 levels quantized skullImage.jpg using imquantize

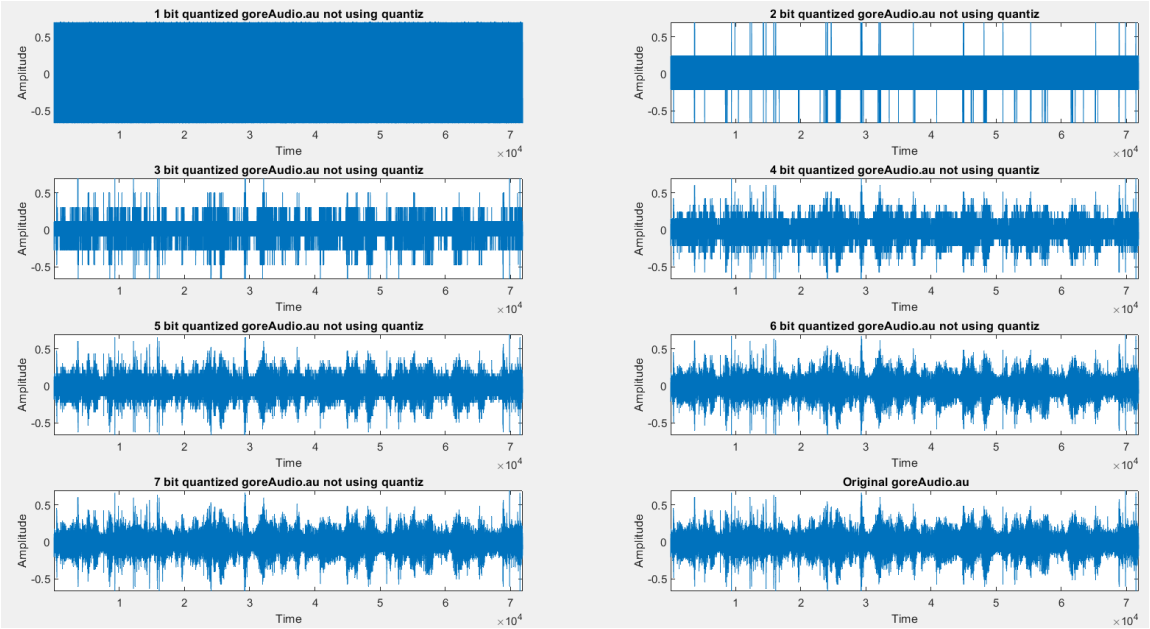
Original skullImage.jpg

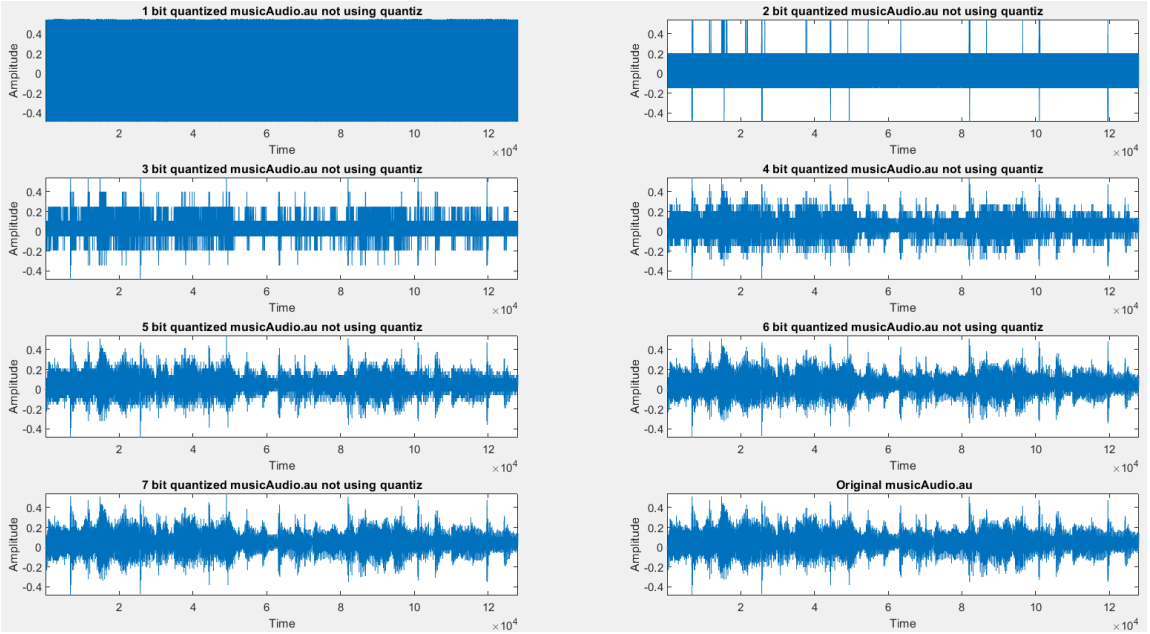


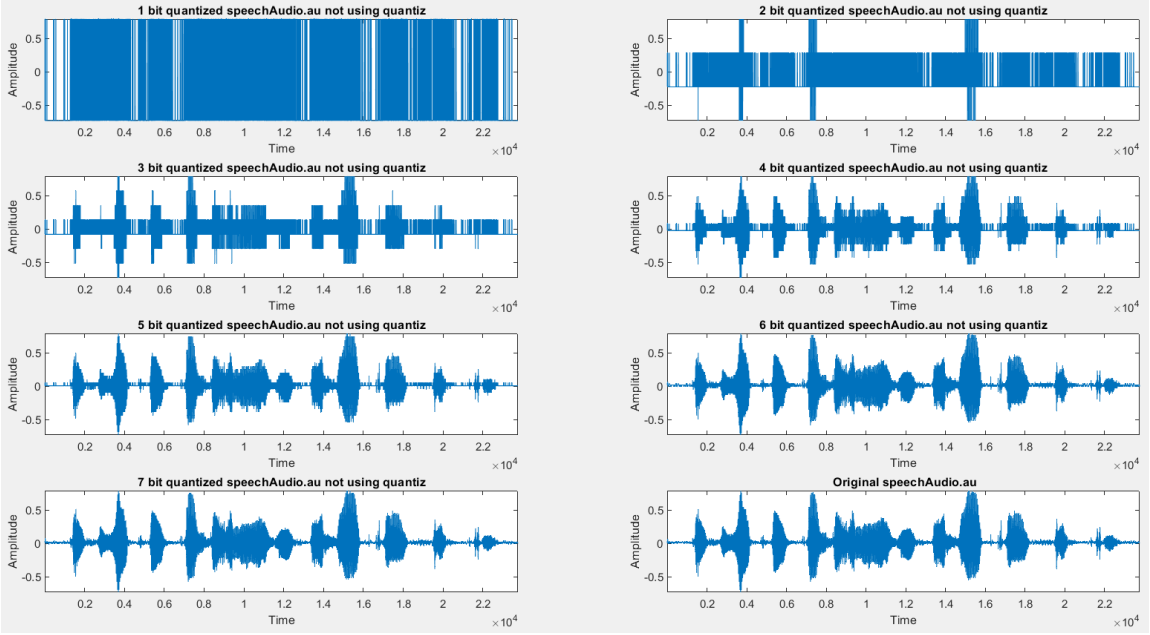
Audio

We quantize different signals a difference bits and we notice how there is more noise after quantizing it when we used fewer bits than when we used more bits to quantize the signal.

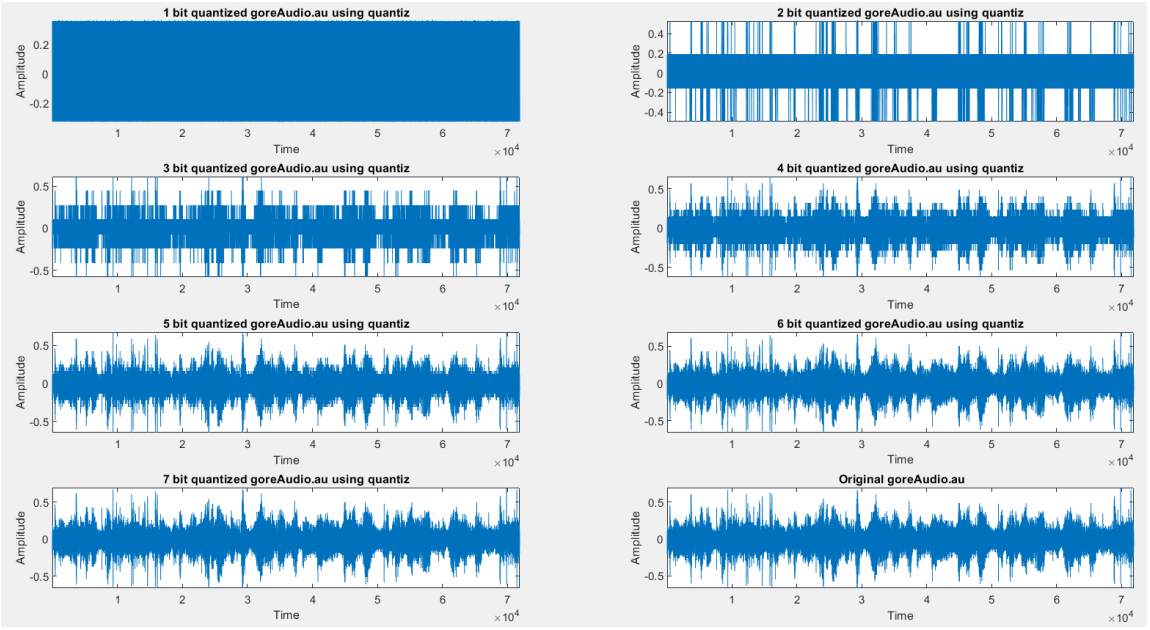
Not Quantizing Audio

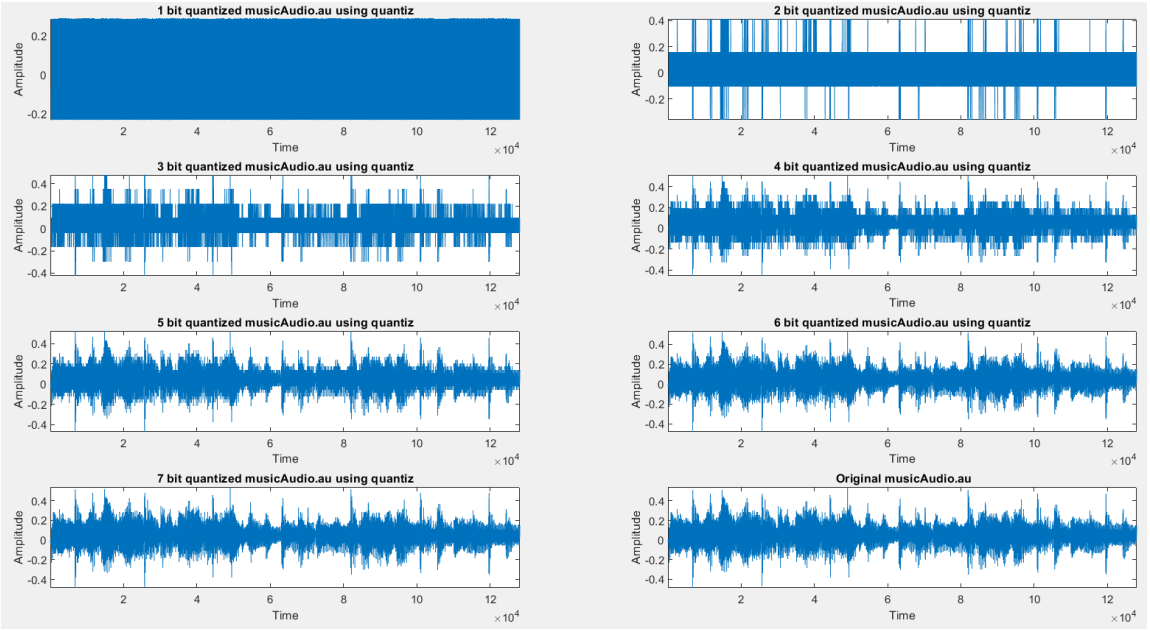


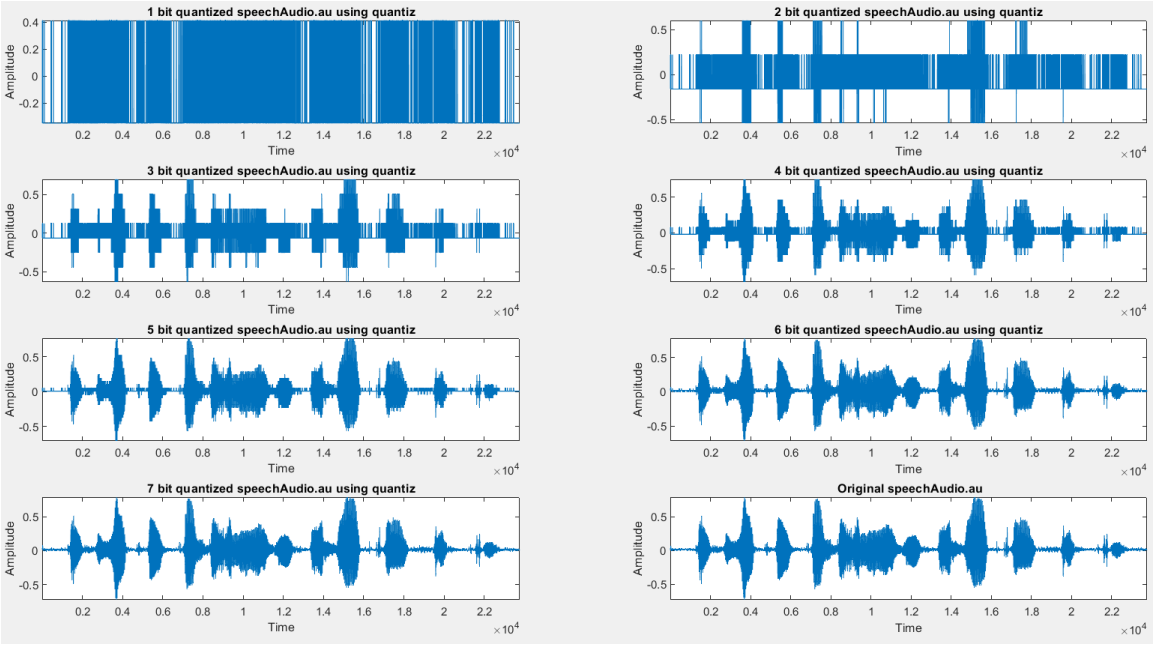




Quantizing Audio

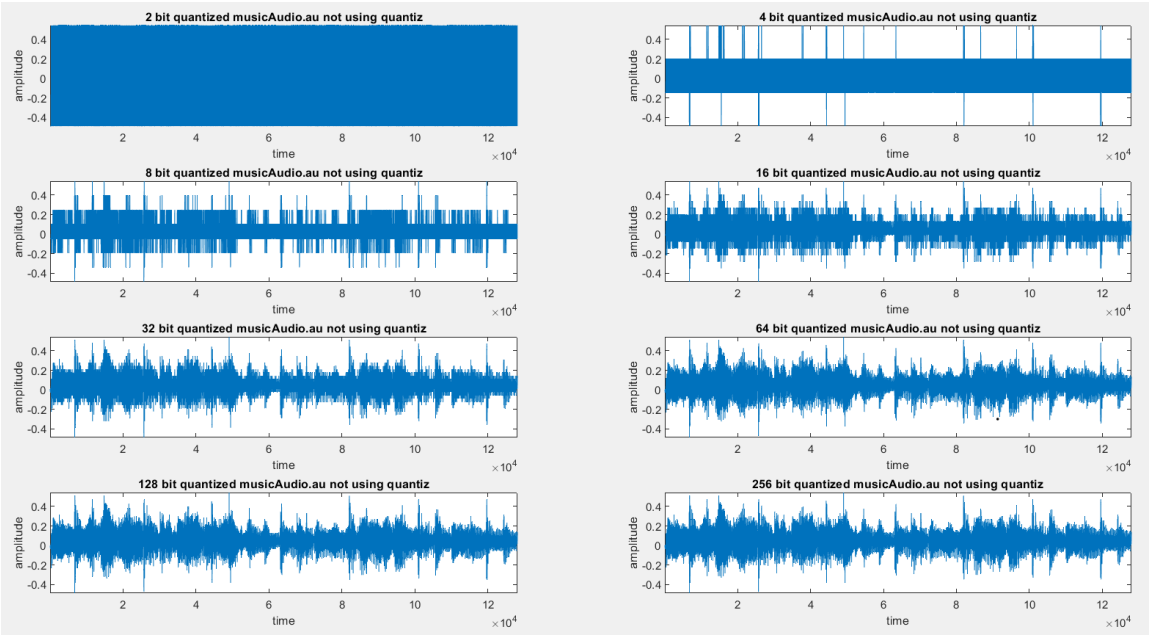


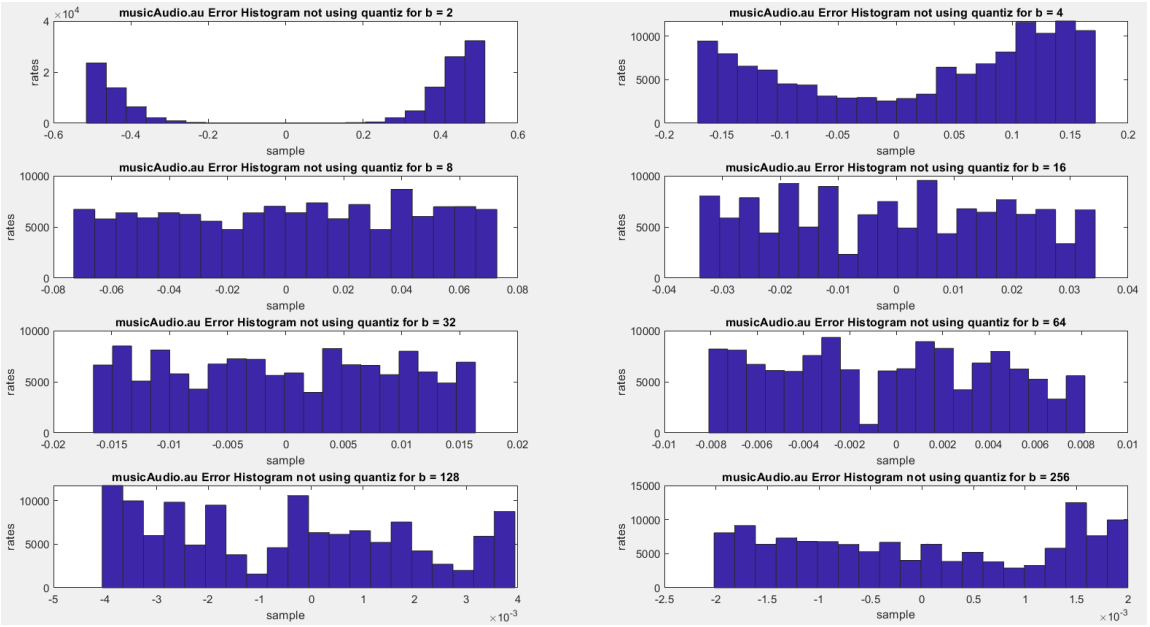


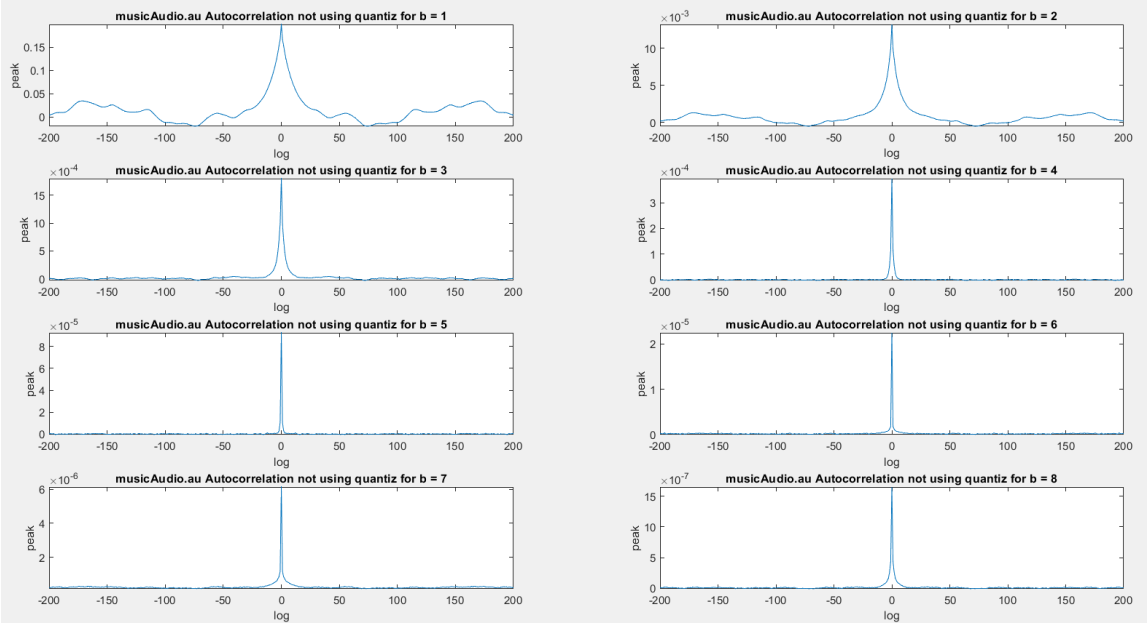


Error Analysis Matlab code

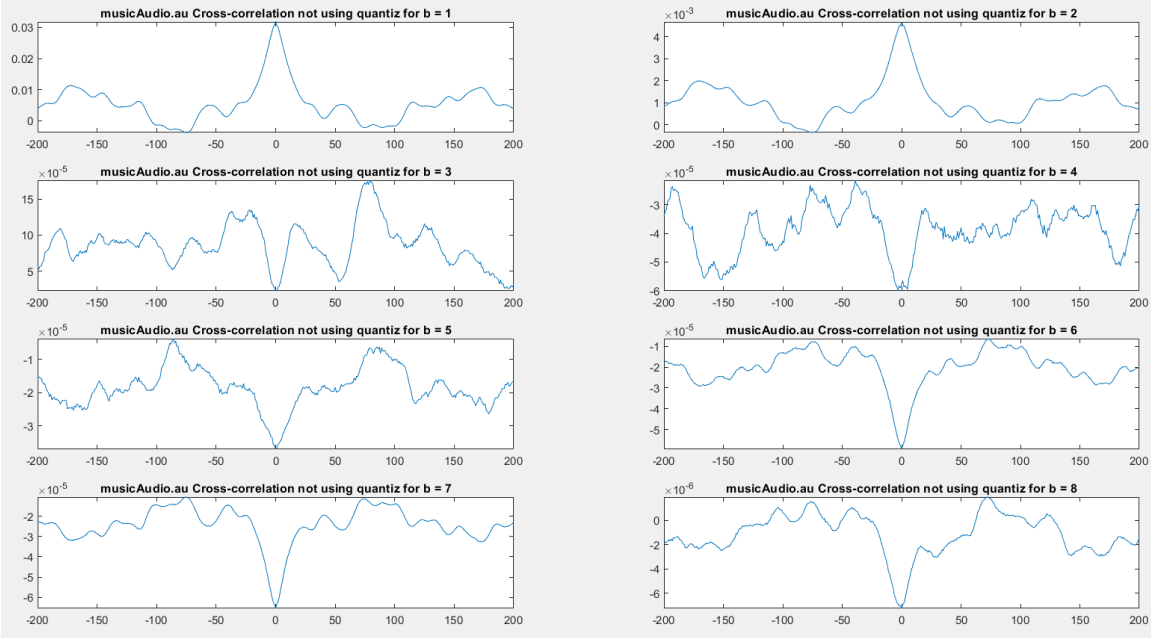
Understanding the error is important because we are able to understand how or analog signal when is a converter from digital back to analog has changed and has been distorted


2 bit quantized musicAudio.au not using quantiz
4 bit quantized musicAudio.au not using quantiz
8 bit quantized musicAudio.au not using quantiz
16 bit quantized musicAudio.au not using quantiz
32 bit quantized musicAudio.au not using quantiz
64 bit quantized musicAudio.au not using quantiz
128 bit quantized musicAudio.au not using quantiz
256 bit quantized musicAudio.au not using quantiz
amplitude
time


musicAudio.au Error Histogram not using quantiz for b = 2
musicAudio.au Error Histogram not using quantiz for b = 4
musicAudio.au Error Histogram not using quantiz for b = 8
musicAudio.au Error Histogram not using quantiz for b = 16
musicAudio.au Error Histogram not using quantiz for b = 32
musicAudio.au Error Histogram not using quantiz for b = 64
musicAudio.au Error Histogram not using quantiz for b = 128
musicAudio.au Error Histogram not using quantiz for b = 256
rates
sample



In this graph above we could tell that when we process the how when we ha



Conclusion

In conclusion, we were able to use Matlab to process both images and sound signal, we were able to notice how quantizing the signal at different when we used a different lower amount of bits the images tend to have more distorting lower ranger of quality then compare when we used more bits to quantize the signal. Because we used fewer bits to quantize the signal we tend to have more error when comparing it to the actual image.

Quantizing is great when it comes to compressing or shirking the signal but we have to be careful when doing this because we could lose the quality of the signal

Reference Matlab code

Lab 6 image

```
clc, clear all, close all;
imgfls_name= {'moonImage.tif','skullImage.jpg'};
% run through the k loop once per file to be quantized
for imgfls_index=1:length(imgfls_name)
[imgfls_content{imgfls_index}, Fs{imgfls_index}] = imread(imgfls_name{imgfls_index});
%
% y is an array with the names of the files
%Best to add files to matlabs current directory but if not, he is an example with a
'windows' path
    %y= imread('H:\username\Documents\ece31011\matlab\lab5_quantization\skull_image.
jpg');
    % Find the maximum value of a 2 dimensional array
    max_value = max(imgfls_content{imgfls_index}(:));
    min_value = min(imgfls_content{imgfls_index}(:));
    range = max_value-min_value;
    % Shift down the data to start at 0 instead of at min_value
    down_shifted=imgfls_content{imgfls_index}-min_value;
    % Number of bits used to encode the quantization levels
    encode_bits = 1:7;
    % Create a figure to plot the quantized images
    fsf(2*imgfls_index-1)=figure(2*imgfls_index-1);
    % Enlarge to fullscreen
    set(fsf(2*imgfls_index-1), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
    % Quantization not using matlab's image imquantize function
    % run through the i loop for every value of b
    for encode_bit_index = 1:length(encode_bits)
        levels=2^encode_bit_index;
        step = range/(levels-1);
        % prepare the data to be quantized so it can be rounded
        % normalize by the step size
        % so every step contains data values of range 1
        down_range=down_shifted/step;
        % round to nearest integer
        % if in the middle go to higher integer
        rounded=round(down_range);
        % un-normalize by the step size
        up_range=rounded*step;
        % add back the minimum value to create the quantized image to N
        % levels
        quantized_image= up_range+min_value;
        % plot the quantized images
        subplot((length(encode_bits)+2)/3,3,encode_bit_index);
        imshow(quantized_image);
        title([num2str(encode_bit_index)  ' bit quantized ' imgfls_name{imgfls_index}
' not using quantiz']);
    end
    % plot the original image
    subplot((length(encode_bits)+2)/3,3,length(encode_bits)+1);
    imshow(imgfls_content{imgfls_index});
```

```
    title(['Original ' imgfls_name{imgfls_index}]);


%     % Quantization using matlab's imquantize function
%     % Note that imquantize/multithresh allows only up to 20 levels.
     for encode_bit_index = 1:4
         numLevels = 2^(encode_bit_index);
         partition = multithresh(imgfls_content{imgfls_index}, numLevels);
         codebook = [partition max(imgfls_content{imgfls_index}(:))];
         [imquantized_image, index] = imquantize(imgfls_content{imgfls_index}, ↙
partition,codebook);
         %plot quantized signal----------------------
         fsf(2*imgfls_index)=figure(2*imgfls_index);
         % Enlarge figure to full screen.
         set(fsf(2*imgfls_index), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
         subplot((length(encode_bits)+2)/3,3,encode_bit_index);
         imshow(imquantized_image);
         title([num2str(numLevels) ' levels quantized ' imgfls_name{imgfls_index} '↙
using imquantize']);
     end
     % plot the original image
     subplot((length(encode_bits)+2)/3,3,4+1);
     imshow(imgfls_content{imgfls_index});
     title(['Original ' imgfls_name{imgfls_index}]);
end
```

Lab 6 audio

```
clc, clear all, close all;
audfls_name = {'goreAudio.au','musicAudio.au','speechAudio.au','boygeorgeAudio.au'};
for imgfls_index=1:length(audfls_name)
[audfls_content{imgfls_index}, Fs{imgfls_index}] = audioread(audfls_name
{imgfls_index});
% to find the maximum value of a 2 dimensional array
    max_value = max(audfls_content{imgfls_index});
    min_value = min(audfls_content{imgfls_index});
    % Shift down the data to start at 0 instead of at min_value
    shift=audfls_content{imgfls_index}- min_value;
    range = max_value - min_value;
    encode_bits = 1:7; % number of bits needed to encode the number of quantization
levels
    %Quantize not using the matlab 'quantiz' function
    fsf(2*imgfls_index-1)=figure(2*imgfls_index-1);
    set(fsf(2*imgfls_index-1), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
    for encode_bit_index = 1:length(encode_bits)
        levels=2^encode_bit_index; %number of quantization levels
        step = range/(levels-1); %step size or interval between two quantization
levels
        % prepare the data to be quantized so it can be rounded
        % normalize by the step size
        % so every step contains data values of range 1
        down_range=shift/step;
        % round to nearest integer
        % if in the middle go to higher integer
        rounded=round(down_range);
        % un-normalize by the step size
        up_range=rounded*step;
        % add back the minimum value
        quantized_audio=up_range+min_value;
        %plot quantized signal----------------------
        subplot((length(encode_bits)+1)/2,2,encode_bit_index);
        plot(quantized_audio);
        title([num2str(encode_bit_index)  ' bit quantized ' audfls_name{imgfls_index}
' not using quantiz' ]);
        xlabel('Time'); ylabel('Amplitude');
        axis tight;
        if (imgfls_index==2 && encode_bit_index==4)%k is the audio file, i is the
number of bits
            soundsc(quantized_audio,16000,16)
            % second argument is fs.
            % may need to change, eg k=2 needs to be played at 16000
        end
    end
    subplot((length(encode_bits)+1)/2,2,length(encode_bits)+1);
    plot(audfls_content{imgfls_index});
    title([' Original ' audfls_name{imgfls_index}]);
    xlabel('Time'); ylabel('Amplitude');
```

```
    axis tight;
    %Quantize using quantiz
    fsf(2*imgfls_index)=figure(2*imgfls_index);
    set(fsf(2*imgfls_index), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
    for encode_bit_index = 1:length(encode_bits)
        levels=2^encode_bit_index;
        minv=min(audfls_content{imgfls_index});
        maxv=max(audfls_content{imgfls_index});
        range=max_value-min_value;
        step=range/(levels);
        lowerpart=step+minv;
        upperpart=maxv-step;
        lowercode=minv+(step/2);
        uppercode=maxv-(step/2);
        [index,quants] = quantiz(audfls_content{imgfls_index},lowerpart:step:
upperpart,lowercode:step:uppercode);
        %plot quantized signal----------------------
        subplot((length(encode_bits)+1)/2,2,encode_bit_index);
        plot(quants);
        title([num2str(encode_bits(encode_bit_index)) ' bit quantized ' audfls_name
{imgfls_index} ' using quantiz' ]);
        xlabel('Time'); ylabel('Amplitude');
        axis tight;
%         {
%         if (k==2 && i==4)%k is the audio file, i is the number of bits
%             soundsc(quants,8000,16)
%         end
%         }
    end
    subplot((length(encode_bits)+1)/2,2,length(encode_bits)+1);
    plot(audfls_content{imgfls_index});
    title([' Original ' audfls_name{imgfls_index}]);
    xlabel('Time'); ylabel('Amplitude');
    axis tight;
end
```

Lab7

```
clc, clear all, close all;
A = {'musicAudio.au' };
for k=1:length(A)
[y{k}, Fs{k}] = audioread(A{k});
end
for k = 1:length(A)
    z = double(y{k});
    max_value = max(y{k});
    min_value = min(y{k});
    range = max_value-min_value;
    shift=z-min_value;
    b = 1:8;% bits to encode from 2 to 256 new quantizing levels
%Quantize not using quantiz
   for i = 1:length(b)
        N=2^i;% number of new quantizing levels (2,4,8,16,32...256)
        delta = range/(N-1);%step size between two levels
        norm=shift/delta;
        rounded=round(norm);
        unnorm=rounded*delta;
        quantized_audio=unnorm+min_value;
        %plot quantized signal----------------------
        fsf(10*k+1)=figure(10*k+1);
        set(fsf(10*k+1), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);%↙
fullscreen
        subplot(length(b)/2,2,i);
        plot(quantized_audio);
        title([num2str(N) ' bit quantized ' A{k} ' not using quantiz' ]);
        xlabel('time'); ylabel('amplitude');
        axis tight;

        %error signal & histogram--------------------
        E=quantized_audio-y{k};
        fsf(10*k+2)=figure(10*k+2);
        set(fsf(10*k+2), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        hist(E,20);
        title([A{k} ' Error Histogram not using quantiz for b = ' num2str(N)]);
        xlabel('sample'); ylabel('rates');
        %autocorrelation for error signal and plot---------
        [r,lags]=xcorr(E,200,'unbiased');
        fsf(10*k+3)=figure(10*k+3);
        set(fsf(10*k+3), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        plot(lags,r);
        title([A{k} ' Autocorrelation not using quantiz for b = ' num2str(b(i))]);
        xlabel('log'); ylabel('peak');
        axis tight;

        %cross-correlation and plot-----------------
```

```
        [c,lags]=xcorr(E,z,200,'unbiased');
        fsf(10*k+4)=figure(10*k+4);
        set(fsf(10*k+4), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        plot(lags,c);
        title([A{k} ' Cross-correlation not using quantiz for b = ' num2str(b(i))]);
%         xlabel('?'); ylabel('?');
        axis tight;
   end
%     %Quantize using quantiz
   for i = 1:length(b)
        levels=2^i;
        minv=min(z);
        maxv=max(z);
        range= maxv-minv;
        delta= range/levels;
        lowerpart= delta+minv;
        upperpart= maxv-delta;
        lowercode= minv+(delta/2);
        uppercode= maxv-( delta/2);
        [index,quants] = quantiz(z,lowerpart:delta:upperpart,lowercode:delta:
uppercode);
        %plot quantized signal----------------------
        fsf(10*k+5)=figure(10*k+5);
        set(fsf(10*k+5), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        plot(quantiz);
        title([num2str(b(i)) ' bit quantized ' A{k} ' using quantiz' ]);
        xlabel('?'); ylabel('?');
        axis tight;
        %error signal & histogram--------------------
        E= quantiz'- fs{k};
        fsf(10*k+6)=figure(10*k+6);
        set(fsf(10*k+6), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        hist(quantiz,20);
        title([A{k} ' Error Histogram using quantiz for b = ' num2str(b(i))]);
        xlabel('?'); ylabel('?');
        %autocorrelation for error signal and plot---------
        [r,lags]=xcorr(E,200,'unbiased');
        fsf(10*k+7)=figure(10*k+7);
        set(fsf(10*k+7), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        plot(lags,r);
        title([A{k} 'Autocorrelation using quantiz for b = ' num2str(b(i))]);
        xlabel('log'); ylabel('peak');
        axis tight;
        %cross-correlation and plot------------------
        [c,lags]=xcorr(E,quants',200,'unbiased');

        fsf(10*k+8)=figure(10*k+8);
        set(fsf(10*k+8), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(length(b)/2,2,i);
        plot(lags,c);
        title([A{k} ' Cross-correlation using quantiz for b = ' num2str(N)]);
        xlabel('log'); ylabel('peak');
        axis tight;
   end
end
```

```
%Power Signal to Noise Ratio (PSNR) Matlab code
clc, clear all, close all;
A = {'goreAudio.au','musicAudio.au','speechAudio.au','boygeorgeAudio.au'};
for k=1:length(A)
[y{k}, Fs{k}] = audioread(A{k});
fsf=figure();
set(fsf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
for k = 1: length(A)
    max_value = max(max(y{k}));
    min_value = min(min(y{k}));
    range = max_value-min_value;
    shift=y{k}-min_value;
    b =1:7;
    %Quantize not using quantiz
   for i = 1:length(b)
        N=2^i;
        delta =range/(N-1);
        norm=shift/delta;
        rounded=round(norm);
        unnorm=norm*shift;
        quantized_audio =unnorm+min_value;
        %error signal---------------------------------
        E=quantized_audio-y{k};
        %signal to noise ratio------------------------
        L1=length(quantized_audio);
        L2=length(E);
        Py=(1/L1)*sum(quantized_audio.^2);
        Pe=(1/L2)*sum(E.^2);
        PSNR=Py/Pe;
        distortion(i)=1/PSNR;
        Fs=8000;
        bit_rate(i)=i*Fs;
    end
    %plot distortion curve not using quantiz
    subplot(4,2,(2*k)-1);
    plot(bit_rate,distortion);
    title(['Distortion Curve not using quantiz on ',A{k}]);
    xlabel('?(bits/s) - ?(sample/s) * ?(bits/sample)'); ylabel('?');
    axis tight;
    %Quantize using quantiz
    for i = 1:length(b)
            levels=2^i;
            minv=min(y{k});
            maxv=max(y{k});
            range=maxv-minv;
            delta=range/(levels);
            lowerpart=delta+minv;
            upperpart=maxv-delta;
            lowercode=minv+(delta/2);
```

```
            uppercode=maxv-(delta/2);
            [index,quants] = quantiz(y{k}, lowerpart:delta:upperpart, lowercode:
delta:uppercode);
            %error signal---------------------------------
            E=quants'-y{k};
            %signal to noise ratio-----------------------
            L1=length(quants');
            L2=length(E);
            Py=(1/L1)*sum(quants'.^2);
            Pe=(1/L2)*sum(E.^2);
            PSNR=Py/PE;
            distortion(i)=1/PSNR;
            Fs=8000;
            bit_rate(i)=i*Fs;
     end
    %plot distortion curve using quantiz-----------------
    subplot(4,2,2*k);
    plot(bit_rate,distortion);
    title(['Distortion Curve using quantiz on ',A{k}]);
    xlabel('?(bits/s) = ?(sample/s) * ?(bits/sample)'); ylabel('?');
    axis tight;
end
end
```