VISVESVARAYA TECHNOLOGICAL UNIVERSITY

MINI PROJECT REPORT ON

"AUDIO COMPRESSION USING WAVELET TRANSFORM"

SUBMITTED BY:

MATAM RISHI (1NH18EC070)

SHIVANI YADAV (1NH18EC103)

NEETHA NATARAJ (1NH18EC077)

NAVODIT TIWARI (1NH18EC074)

Under the guidance of

Prof. Dr. REEMA SHARMA

Senior Assistant Professor, Dept. of ECE, NHCE, Bengaluru.

NEW HORIZON COLLEGE OF ENGINEERING

(ISO-9001:2000 certified, Accredited by NAAC 'A', Autonomous college permanently affiliated to VTU), Outer Ring Road, Panathur Post, Near Marathalli, Bengaluru-560103

NEW HORIZON COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the mini project work entitled "**AUDIO COMPRESSION USING WAVELET TRANSFORM**" carried out by NEETHA NATARAJ **(1NH18EC077), SHIVANI YADAV(1NH18EC103), MATAM RISHI (1NH18EC070), NAVODIT TIWARI (1NH18EC074),** bonafide students of Electronics and Communication Department, New Horizon College of Engineering, Bangalore.

The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the said degree.

Project Guide HOD ECE

---------- ----------

External Viva

Name of Examiner Signature with Date

1.

2.

ACKNOWLEDGEMENT

The satisfaction that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement helped us succeed.

We thank **Dr. Mohan Manghnani**, Chairman of **New Horizon Educational Institution**, for providing necessary infrastructure and creating good environment.

We also record here the constant encouragement and facilities extended to us by **Dr. Manjunatha**, Principal, NHCE and **Dr. Sanjeev Sharma**, head of the department of Electronics and Communication Engineering. We extend sincere gratitude to them.

We sincerely acknowledge the encouragement, timely help and guidance to us by our beloved guide **Prof. Dr. Reema Sharma** to complete the project within stipulated time successfully.

Finally, a note of thanks to the teaching and non-teaching staff of electronics and communication department for their co-operation extended to us, who helped us directly or indirectly in this successful completion of mini project.

Neetha Nataraj (1NH18EC077)

Shivani Yadav(1NH18EC103)

Matam Rishi (1NH18EC070)

Navodit Tiwari (1NH18EC074)

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Audio Compression

Compression reduces the dynamic range of your recording by bringing down the level of the loudest parts, meaning the loud and quiet parts are now closer together in volume and the natural volume variations are less obvious.

The audio compressor unit can then boost the overall level of this compressed signal. So the end result is that the quieter parts sound like they've been boosted in volume to be closer to the louder parts.

The dynamic volume changes of a recording are now under more control, and a knock-on effect is that the overall level of the compressed recording can be increased inside your mix. The recording will also sit inside your whole mix much more easily.

1.2 METHODS USED FOR COMPRESSION

1.2.1 HAAR TRANSFORM

A Haar wavelet is the simplest type of wavelet. In discrete form, Haar wavelets are related to a mathematical operation called the Haar transform. The Haar transform serves as a prototype for all other wavelet transforms. Studying the Haar transform in detail will provide a good foundation for understanding the more sophisticated wavelet transforms which we shall describe in the next chapter. In this chapter we shall describe how the Haar

transform can be used for compressing audio signals and for removing noise. Our discussion of these applications will set the stage for the more powerful wavelet transforms to come and their applications to these same problems. One distinctive feature that the Haar transform enjoys is that it lends itself Easily to simple hand calculations. We shall illustrate many concepts by both simple hand calculations and more involved computer computations.

1.2.2 DAUBENCHES TRANSFORM

The Daubeches wavelet transforms are defined in the same way as the Haar wavelet transform—by computing running averages and differences via scalar products with scaling signals and wavelets the only difference between them consists in how these scaling signals and wavelets are defined. For the Daubeches wavelet transforms, the scaling signals and wavelets have slightly longer supports, i.e., they produce averages and differences using just a few more values from the signal. This slight change, however, provides a tremendous improvement in the capabilities of these new transforms. They provide us with a set of powerful tools for performing basic signal processing tasks. These tasks include compression and noise removal for audio signals and for images, and include image enhancement and signal recognition.

CHAPTER 2

LITERATURE SURVEY

1.1 .WAV FILES

A WAV file is a raw audio format created by Microsoft and IBM. The format uses containers to store audio data, track numbers, sample rate, and bit rate. WAV files are uncompressed lossless audio and as such can take up quite a bit of space, coming in around 10 MB per minute with a maximum file size of 4 GB.

WAV file formats use containers to contain the audio in raw and typically uncompressed “chunks” using the Resource Interchange File Format (RIFF). This is a common method Windows uses for storing audio and video files— like AVI— but can be used for arbitrary data as well.

WAV files are generally going to be much larger than other popular audio file types, like MP3, due to the fact they are typically uncompressed (compression is supported, though). Because of this, they’re mainly used in the professional music recording industry to retain the maximum quality of audio.

A WAV file is an audio file saved in the WAVE format, which is a standard digital audio file format utilized for storing waveform data. WAV files may contain audio recordings with different sampling

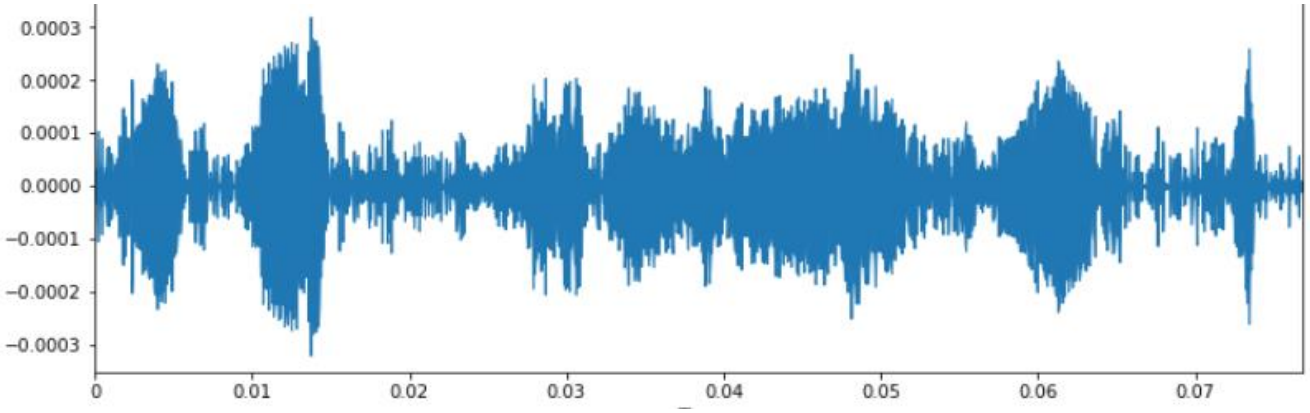

Fig 2.1: A sample audio waveform in wav

rates and bitrates but are often saved in a 44.1 KHz, 16-bit, stereo format, which is the standard format used for CD audio.

Fig 2.2: Icon of a .wav format file

WAV files are similar to .AIF files, which are saved in the Audio Interchange File Format (AIFF) and are more common on Macintosh systems. However, WAV files are more often saved and played on Windows computers rather than Macs.

There are a large number of programs that support WAV files on Windows, macOS, and Linux systems. Some of them include bundled programs, such as Microsoft Windows Media Player and Apple iTunes, and third-party applications, such as VideoLAN VLC media player and Eltima Elmedia Player.

1.2 STEPS TAKEN TO ATTAIN COMPRESSION

The main goal of the algorithm presented in paper is to compress high quality audio maintaining transparent quality at low bit rates. In order to do this, the authors explored the usage of wavelets instead of the traditional Modified Discrete Cosine Transform (MDCT). Several steps are considered to achieve this goal.

• Design a wavelet representation for audio signals.

• Design a psychoacoustic model to perform perceptual coding and adapt it to the wavelet representation.

• Reduce the number of the non-zero coefficients of the wavelet representation and perform quantization over those coefficients.

• Perform extra compression to reduce redundancy over that representation

• Transmit or store the steam of data. Decode and reconstruct.

• Evaluate the quality of the compressed signal.

CHAPTER 3

EXISTING SYSTEM AND PROBLEM STATEMENT

3.1 EXISTING SYSTEM

There are currently various methods of audio compression that are available. They can be broadly classified into lossy audio compression and lossless audio compression. Lossy compression algorithms include Linear Predictive Coding (LPC), Code Excited Linear Predictor (CELP), Adaptive Differential Pulse Code Modulation (ADPCM), and so on. Lossless compression algorithms include Audio Lossless Coding, Adaptive Transform Acoustic Coding, Haar and Daubenches algorithms, and so on.

3.2 PROBLEM STATEMENT

To compress an input audio signal, using lossless compression algorithms.

CHAPTER 4

PROPOSED SYSTEM

The project in discussion aims to perform lossless audio compression in MATLAB using wavelet transform, by using two algorithms: **Haar** and **Daubenches**. We aim to be able to compress the audio signal with a compression ratio of **two**, that is, half the size of the original audio, with as trivial loss of data as possible.

The parameters taken into consideration by the program are:

a. **Peak Signal-To-Noise Ratio (PSNR):** The ratio of the power of the signal to the power of the unwanted noise signal that has an effect on the fidelity of the signal.

   PSNR = $10 \log_{10} \frac{NX^2}{\|x-r\|^2}$; where:

   N represents the length of the modified signal,

   X represents the maximum absolute square of the input signal,

   $||x\text{-}r^2||$ represents the difference in energy between the input and compressed audio signals.

b. **Root-Mean-Square Error (RMSE):** This parameter is used to measure the deviation of a signal.

   RMSE = $\sqrt{\frac{\{x(n)-r(n)\}^2}{\{x(n)-\mu_A(n)\}^2}}$; where:

   x(n) represents the input audio signal,

   r(n) represents the compressed audio signal,

   μ(n) represents the mean of the given audio signal.

c. **Ratio of Compression:** The ratio of the length of the input audio signal to the length of the length of the compressed audio signal.

   C = $\frac{Length\,(x(n))}{Length\,(cWc)}$; where:

   x(n) represents the input audio signal,

   cW_c represents the Wavelet Transform Vector.

4.1 HAAR ALGORITHM-

4.1.1 Block Diagram:

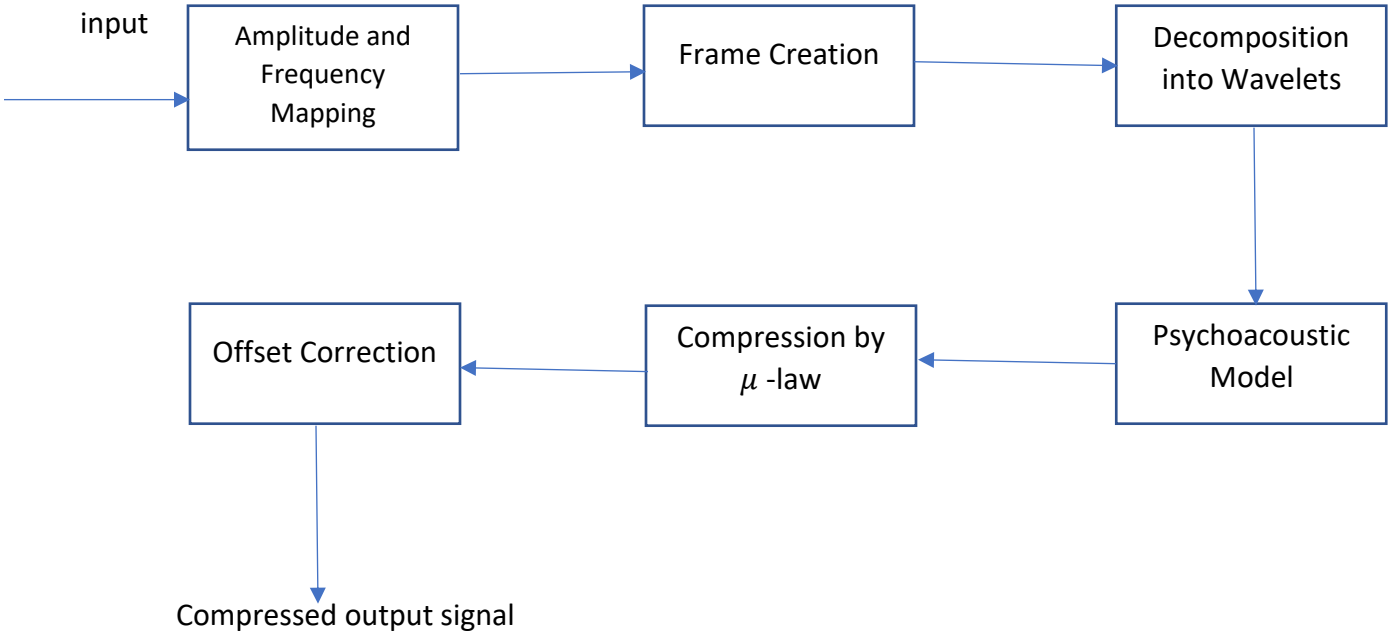

Fig 4.1: Block Diagram of Haar Algorithm

4.1.2 Algorithm steps:

1. Input audio is selected and its original size is noted.
2. The amplitude and frequency of the signal is mapped.
3. Frames, or audio samples, are created. An audio sample can be defined as the smallest quantizable unit of digital audio.
4. The signal spectrum is decomposed into wavelets.
5. A psychoacoustic model of the audio is generated. A psychoacoustic analyzes the audio signal and computes the amount of noisemaking as a function of frequency.
6. The spectrum is inspected and tone maskers are spotted. Tone masking occurs when the perception of one sound is affected by the presence of another sound.
7. The μ -Law of Compression is applied. The law is given by:

   $y = \frac{V \log(1+\mu|x|/V)}{\log(1+\mu)} \text{sgn}(x)$; where, V is the maximum value of the signal, and μ is the μ law parameter. Sgn(X) represents the signum function.

8. Any audio offset is found and corrected.
9. The wave is rewritten.
10. The size of the output wave is noted.

4.1.3 MATLAB code :

```
function varargout = AudioCompression(varargin)
% AUDIOCOMPRESSION MATLAB code for AudioCompression.fig
%      AUDIOCOMPRESSION, by itself, creates a new AUDIOCOMPRESSION or
raises the existing
%      singleton*.
%
%      H = AUDIOCOMPRESSION returns the handle to a new
AUDIOCOMPRESSION or the handle to
%      the existing singleton*.
%
%      AUDIOCOMPRESSION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in AUDIOCOMPRESSION.M with the given
input arguments.
%
%      AUDIOCOMPRESSION('Property','Value',...) creates a new
AUDIOCOMPRESSION or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before AudioCompression_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to AudioCompression_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
```

```
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AudioCompression

% Last Modified by GUIDE v2.5 21-Nov-2014 17:35:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AudioCompression_OpeningFcn,
...
                   'gui_OutputFcn',  @AudioCompression_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before AudioCompression is made visible.
Function AudioCompression_OpeningFcn(hObject, eventdata, handles,
varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AudioCompression (see
VARARGIN)

% Choose default command line output for AudioCompression
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AudioCompression wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
Function varargout = AudioCompression_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
Function pushbutton1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
%guidata(hObject,handles)
file_name=uigetfile({'*.wav'},'Select an Audio File');
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs☹length(x)-1)/Fs;
set(handles.text2,'string',Size);
%plot(t,x);
axes(handles.axes3) % Select the proper axes
plot(t,x)
set(handles.axes3,'XminorTick','on')
grid on

% --- Executes on button press in pushbutton2.
Function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
if(~ischar(file_name))
   errordlg('Please select Audio first');
 else
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs☹length(x)-1)/Fs;
wavelet='haar';
```

```
level=5;
frame_size=2048;
psychoacoustic='on '; %if it is off it uses 8 bits/frame as default
wavelet_compression = 'on ';
heavy_compression='off';
compander='on ';
quantization ='on ';

% ENCODER

step=frame_size;
N=ceil(xlen/step);
%computational variables
Cchunks=0;
Lchunks=0;
Csize=0;
PERF0mean=0;
PERFL2mean=0;
n_avg=0;
n_max=0;
n_0=0;
n_vector=[];
for i=1:1:N
if (i==N);
frame=x([(step*(i-1)+1):length(x)]);
else
frame=x([(step*(i-1)+1):step*i]);
end
%wavelet decomposition of the frame
[C,L] = wavedec(frame,level,wavelet);
%wavelet compression scheme
if wavelet_compression=='on '
[thr,sorh,keepapp] = ddencmp('cmp','wv',frame);
```

```
if heavy_compression == 'on '
thr=thr*10^6;
end
[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('gbl',C, L,
wavelet,level,thr,sorh,keepapp);
C=CXC;
L=LXC;
PERF0mean=PERF0mean + PERF0;
PERFL2mean=PERFL2mean+PERFL2;
end
%Psychoacoustic model
if psychoacoustic=='on '
P=10.*log10((abs(fft(frame,length(frame)))).^2);
Ptm=zeros(1,length(P));
%Inspect spectrum and find tones maskers
for k=1:1:length(P)
if ((k<=1) | (k>=250))
bool = 0;
elseif ((P(k)<P(k-1)) | (P(k)<P(k+1))),
bool = 0;
elseif ((k>2) & (k<63)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)));
elseif ((k>=63) & (k<127)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)) & (P(k)>(P(k-3)+7)) &
(P(k)>(P(k+3)+7)));
elseif ((k>=127) & (k<=256)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)) & (P(k)>(P(k-3)+7)) &
(P(k)>(P(k+3)+7)) & (P(k)>(P(k-4)+7)) & (P(k)>(P(k+4)+7))
&(P(k)>(P(k-5)+7)) & (P(k)>(P(k+5)+7)) & (P(k)>(P(k-6)+7))
&(P(k)>(P(k+6)+7)));
else
bool = 0;
end
if bool==1
```

```
Ptm(k)=10*log10(10.^(0.1.*(P(k-
1)))+10.^(0.1.*(P(k)))+10.^(0.1.*P(k+1)));
end
end
sum_energy=0;
for k=1:1:length(Ptm)
sum_energy=10.^(0.1.*(Ptm(k)))+sum_energy;
end
E=10*log10(sum_energy/(length(Ptm)));
SNR=max(P)-E;
n=ceil(SNR/6.02);
if n<=3
n=4;
n_0=n_0+1;
end
if n>=n_max
n_max=n;
end
n_avg=n+n_avg;
n_vector=[n_vector n];
end
%Compander(compressor)
if compander=='on '
Mu=255;
C = compand(C,Mu,maxI,'mu/compressor');
end
%Quantization
if quantization=='on '
if psychoacoustic=='off'
n=8;
end
partition = [minI☹(maxI-minI)/2^n):maxI];
codebook = [1 minI☹(maxI-minI)/2^n):maxI];
```

```
[index,quant,distor] = quantiz(C,partition,codebook);
%find and correct offset
offset=0;
for j=1:1:N
if C(j)==0
offset=-quant(j);
break;
end
end
quant=quant+offset;
C=quant;
end
%Put together all the chunks
Cchunks=[Cchunks C];
Lchunks=[Lchunks L];
Csize=[Csize lengthI];
Encoder = round((i/N)*100); %indicator of progess
end
Cchunks=Cchunks(2:length(Cchunks));
%wavwrite(Cchunks,Fs,bits,'output1.wav')
Csize=[Csize(2) Csize(N+1)];
Lsize=length(L);
Lchunks=[Lchunks(2:Lsize+1) Lchunks((N-1)*Lsize+1:length(Lchunks))];
PERF0mean=PERF0mean/N; %indicator
PERFL2mean=PERFL2mean/N;%indicator
n_avg=n_avg/N;%indicator
n_max;%indicator
end_of_encoder='done';
xdchunks=0;
for i=1:1:N;
if i==N;
Cframe=Cchunks([((Csize(1)*(i-1))+1):Csize(2)+(Csize(1)*(i-1))]);
%Compander (expander)
if compander=='on '
```

```
if max(Cframe)==0
else
Cframe = compand(Cframe,Mu,max(Cframe),'mu/expander');
end
end
xd = waverec(Cframe,Lchunks(Lsize+2:length(Lchunks)),wavelet);
else
Cframe=Cchunks([((Csize(1)*(i-1))+1):Csize(1)*i]);
%Compander (expander)
if compander=='on '
if max(Cframe)==0
else
Cframe = compand(Cframe,Mu,max(Cframe),'mu/expander');
end
end
xd = waverec(Cframe,Lchunks(1:Lsize),wavelet);
end
xdchunks=[xdchunks xd];
Decoder = round((i/N)*100); %indicator of progess
end
xdchunks=xdchunks(2:length(xdchunks));
%distorsion = sum((xdchunks-x').^2)/length(x)
end_of_decoder='done';
%creating audio files with compressed schemes
wavwrite(xdchunks,Fs,bits,'output1.wav');
end_of_writing_file='done';%indicator of progess;
[x,Fs,bits] = wavread('output1.wav');
fileinfo = dir('output1.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
set(handles.text3,'string',Size)
xlen=length(x);
t=0:1/Fs☹length(x)-1)/Fs;
axes(handles.axes4) % Select the proper axes
```

```
plot(t,xdchunks)
set(handles.axes4,'XminorTick','on')
grid on

[y1,fs1, nbits1,opts1]=wavread(file_name);
[y2,fs2, nbits2,opts2]=wavread('output1.wav');
[c1x,c1y]=size(y1);
[c2x,c2y]=size(y1);
if c1x ~= c2x
    disp('dimeonsions do not agree');
 else
 R=c1x;
 C=c1y;
  err = (sum(y1(2)-y2).^2)/(R*C);
 MSE=sqrt(err);
 MAXVAL=255;
  PSNR = 20*log10(MAXVAL/MSE);
  MSE= num2str(MSE);
  if(MSE > 0)
  PSNR= num2str(PSNR);
  else
PSNR = 99;
end
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
fileinfo1 = dir('output1.wav');
SIZE1 = fileinfo1.bytes;
Size1 = SIZE1/1024;

CompressionRatio = Size/Size1;

  set(handles.text14,'string',PSNR)
```

```
  set(handles.text16,'string',MSE)
  set(handles.text17,'string',CompressionRatio)

end

 end


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double



% --- Executes during object creation, after setting all properties.
Function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
If ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double



% --- Executes during object creation, after setting all properties.
Function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
If ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

4.2 DAUBENCHES ALGORITHM

4.2.1 Block Diagram:

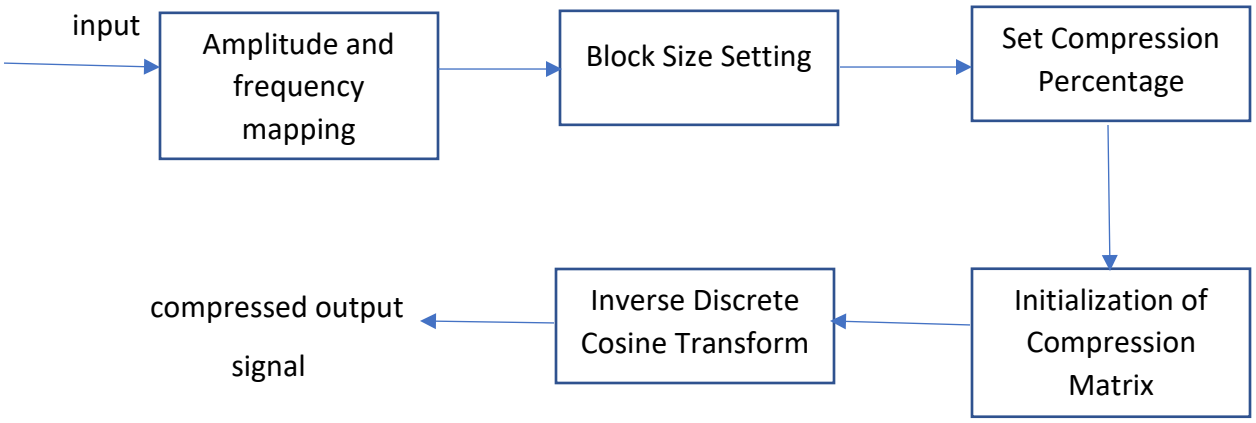

Fig 4.2: Block diagram of Daubenches algorithm

4.2.2 Algorithm Steps:

1. Input audio is selected and its original size is noted.
2. The amplitude and frequency of the signal is mapped.
3. A block size is selected. Block size refers to the sample delay time between the input and output signals.
4. The compression percentage is set.
5. The compression matrix is initialized.
6. Compression is performed using Inverse Discrete Cosine Transform. It is given by:

$$x(n) = \frac{2}{N} \sum_{k=0}^{N-1} e(k) X(k) \cos\left[\frac{(2n+1)\pi k}{2N}\right], \; n = 0, 1, \cdots, N-1,$$

7. The size of the compressed audio signal is noted.

4.2.3 MATLAB code:

```
function varargout = AudioCompression2(varargin)
% AUDIOCOMPRESSION2 MATLAB code for AudioCompression2.fig
%      AUDIOCOMPRESSION2, by itself, creates a new AUDIOCOMPRESSION2
or raises the existing
%      singleton*.
%
%      H = AUDIOCOMPRESSION2 returns the handle to a new
AUDIOCOMPRESSION2 or the handle to
%      the existing singleton*.
%
%      AUDIOCOMPRESSION2('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in AUDIOCOMPRESSION2.M with the given
input arguments.
%
%      AUDIOCOMPRESSION2('Property','Value',...) creates a new
AUDIOCOMPRESSION2 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before AudioCompression2_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to AudioCompression2_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
AudioCompression2
```

```
% Last Modified by GUIDE v2.5 21-Nov-2014 18:23:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AudioCompression2_OpeningFcn,
...
                   'gui_OutputFcn',  @AudioCompression2_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before AudioCompression2 is made visible.
function AudioCompression2_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to AudioCompression2 (see
VARARGIN)

% Choose default command line output for AudioCompression2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AudioCompression2 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = AudioCompression2_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
```

```
%guidata(hObject,handles)
file_name=uigetfile({'*.wav'},'Select an Audio File');
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text12,'string',Size);
%plot(t,x);
axes(handles.axes1) % Select the proper axes
plot(t,x)
set(handles.axes1,'XMinorTick','on')
grid on


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
if(~ischar(file_name))
   errordlg('Please select Audio first');
else
[Data,Fs,bits] = wavread(file_name);
[Data, Fs, bits] = wavread('Windows XP Startup.wav');


%chosing a block size
windowSize = 8192;


%changing compression  percentages
samplesHalf = windowSize / 2;
samplesQuarter = windowSize / 4;
```

```
samplesEighth = windowSize / 8;

%initializing compressed matrice
DataCompressed2 = [];
DataCompressed4 = [];
DataCompressed8 = [];

%actual compression
for i=1:windowSize:length(Data)-windowSize
    windowDCT = dct(Data(i:i+windowSize-1));
    DataCompressed2(i:i+windowSize-1) =
idct(windowDCT(1:samplesHalf), windowSize);
    DataCompressed4(i:i+windowSize-1) =
idct(windowDCT(1:samplesQuarter), windowSize);
    DataCompressed8(i:i+windowSize-1) =
idct(windowDCT(1:samplesEighth), windowSize);
end

wavwrite(DataCompressed2,Fs,bits,'output3.wav')
[x,Fs,bits] = wavread('output3.wav');
fileinfo = dir('output3.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text14,'string',Size);
%plot(t,x);
axes(handles.axes2) % Select the proper axes
plot(t,x)
set(handles.axes2,'XMinorTick','on')
grid on

wavwrite(DataCompressed4,Fs,bits,'output4.wav')
```

```
[x,Fs,bits] = wavread('output4.wav');
fileinfo = dir('output4.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text16,'string',Size);
%plot(t,x);
axes(handles.axes3) % Select the proper axes
plot(t,x)
set(handles.axes3,'XMinorTick','on')
grid on

wavwrite(DataCompressed8,Fs,bits,'output5.wav')
[x,Fs,bits] = wavread('output5.wav');
fileinfo = dir('output5.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text18,'string',Size);
%plot(t,x);
axes(handles.axes4) % Select the proper axes
plot(t,x)
set(handles.axes4,'XMinorTick','on')
grid on



[y1,fs1, nbits1,opts1]=wavread(file_name);
[y2,fs2, nbits2,opts2]=wavread('output3.wav');
[c1x,c1y]=size(y1);
[c2x,c2y]=size(y1);
if c1x ~= c2x
```

```
    disp('dimeonsions do not agree');
 else
 R=c1x;
 C=c1y;
  err = (sum(y1(2)-y2).^2)/(R*C);
 MSE=sqrt(err);
 MAXVAL=255;
   PSNR = 20*log10(MAXVAL/MSE);
  MSE= num2str(MSE);
if(MSE > 0)
  PSNR= num2str(PSNR);
   else
PSNR = 99;
end
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
fileinfo1 = dir('output3.wav');
SIZE1 = fileinfo1.bytes;
Size1 = SIZE1/1024;

CompressionRatio = Size/Size1;


  set(handles.text21,'string',PSNR)
  set(handles.text23,'string',MSE)
  set(handles.text24,'string',CompressionRatio)
end
```

CHAPTER 5

SOFTWARE SPECIFICATIONS

The software platform used to write and execute the source code is **MATLAB R2020.** MATLAB is a laboratory of matrices, a numerical computing and simulation environment, which is multi-paradigm programming language, developed and owned by MathWorks. It's a high-performance language that coalesces computation, visualization and programming in a user-friendly set-up where problems and their solutions are expressed as known Mathematical equations. The most common uses of the software are as follows:

1. Mathematics and it's computation
2. Development of various algorithms
3. Modelling of systems, their simulation and final prototyping
4. Analysis of data and visualization
5. Scientific and engineering based graphical solutions
6. Development of applications, like Graphical User Interfaces

THE MATLAB SYSTEM:

The MATLAB system comprises of mainly five parts:

1. **MATLAB** LANGUAGE

   The MATLAB language is a high-level language construct with various programming features, like control flow statements, object-oriented programming features, methods, data structures and so on. It allows the user to create a wide range of programs, that can be considerably complex.
2. THE **MATLAB** WORKING ENVIRONMENT

The MATLAB user has a set of tools and special facilities at their disposal, which can be used for managing workspace, handling data, developing systems, debugging code and other such applications.

3. HANDLE GRAPHICS

   The MATLAB graphics system contains high-level commands for both 2D and 3D data visualization, animation, image processing and presentation. It also consists of low-level commands that enable the user to custom-make the appearance of data or build GUI's for their MATLAB applications.

4. THE **MATLAB** MATHEMATICAL FUNCTION LIBRARY

   MATLAB comprises of a huge collection of inbuilt algorithms for complex mathematical computations, like matrix functions, Fourier Transformations and so on. This vastly decreases the length of the code and makes computation and debugging much faster and easier.

5. THE **MATLAB** APPLICATION PROGRAM INTERFACE (API)

   This MATLAB feature is a library that allows the user to write code in other programming languages, like C, that can interact with MATLAB. They can have various blocks of code that can be linked with the MATLAB code.

CHAPTER 6

RESULTS AND DISCUSSIONS

MATLAB code for audio compression of a given input signal was written and executed successfully.

Following is the output parameters observed for the project in discussion:

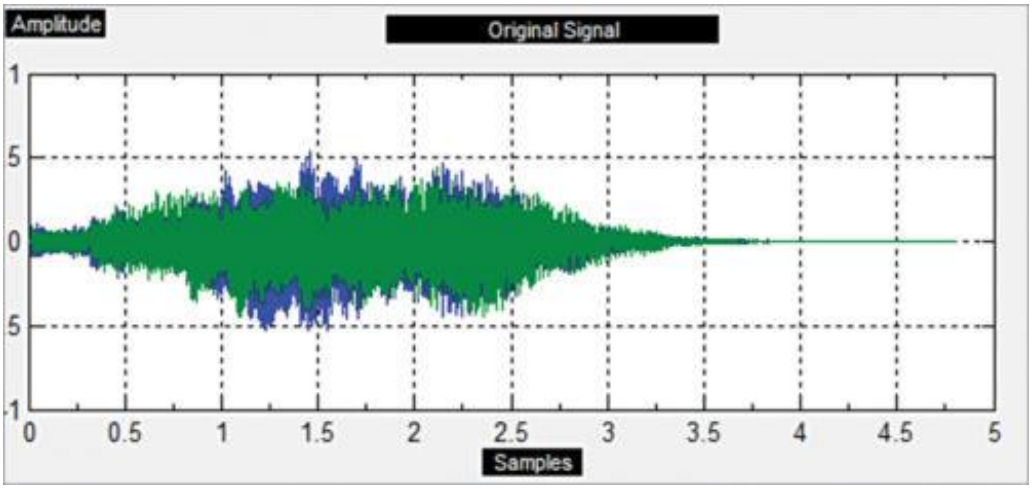

Fig 6.1: Original Signal

1. HAAR WAVELET ALGORITHM:

1. Peak Signal-To-Noise Ratio (PSNR) = 82.84
2. Mean Square Error (MSE) = 8.81
3. Compression Ratio = 1.99

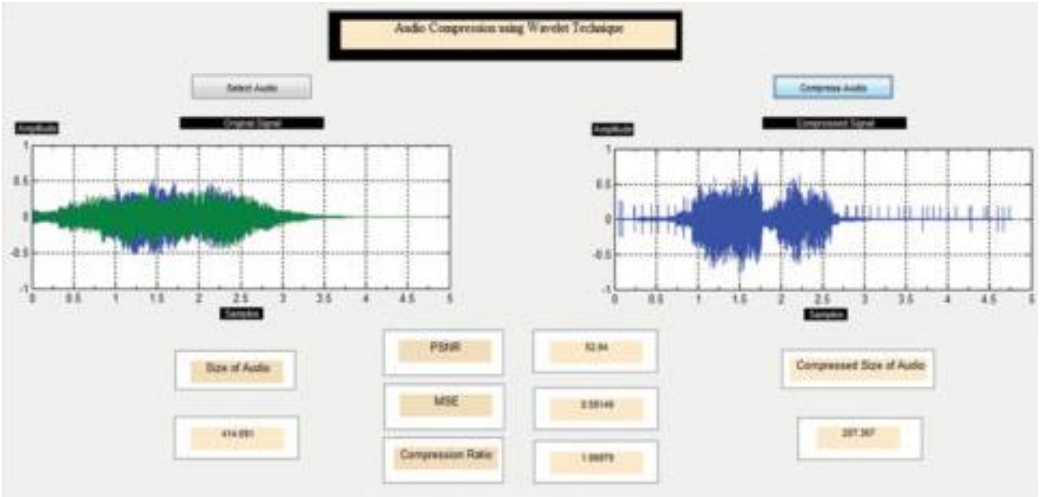

Fig 6.2: Program Output for Haar Wavelet Algorithm

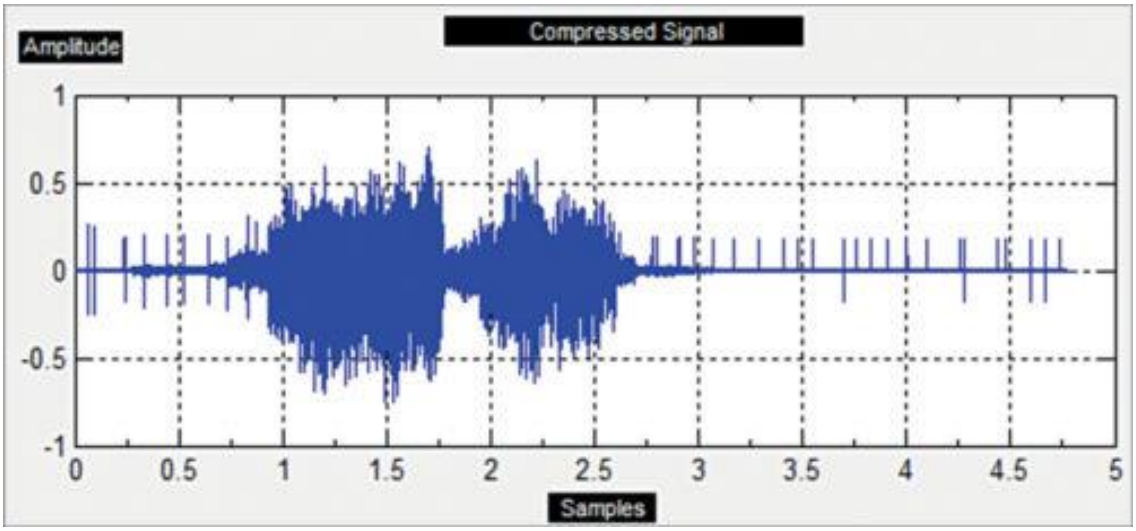

Fig 6.3: Haar Wavelet Decomposed Signal

2. DAUBENCHES WAVELET ALGORITHM:

1. Peak Signal-To-Noise Ratio (PSNR) = 88.47
2. Mean Square Error (MSE) = 8.84
3. Compression Ratio = 2.14

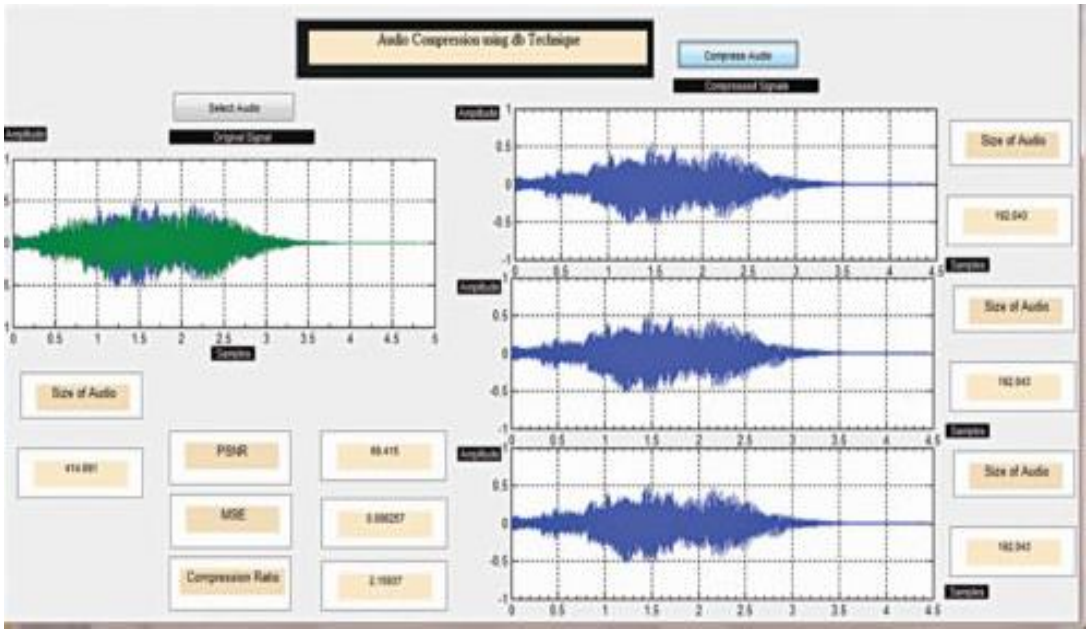

Fig 6.4: Program Output for Daubenches Wavelet Algorithm for Discrete Cosine Transform window size 2, 4 and 6

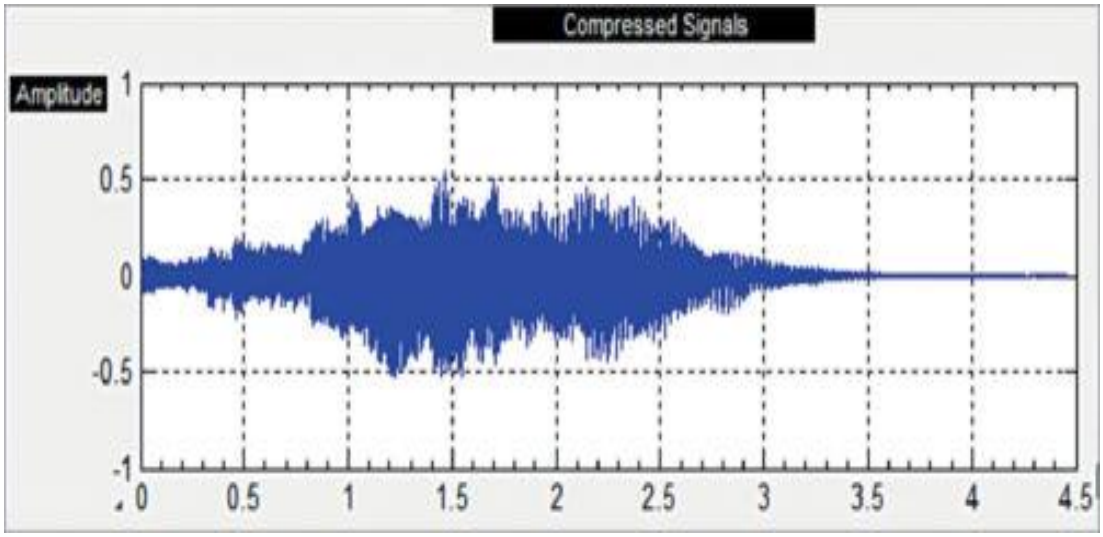

Fig 6.5: Daubenches Decomposed Output Signal

CHAPTER 7

ADVANTAGES AND APPLICATIONS

Audio compression proves to be one of the most effective and popular techniques of data modification, in communications. Almost every mode of communication uses compression techniques, owing to its various advantages. They are as follows:

1. **Reduction in size:** The most useful and obvious application of audio compression is the reduction in its file size. The project in discussion compresses input audio by a factor of **two**, which implies that it reduces the storage required to store the file by half. Hence, storage space is greatly saved.
2. **Faster Transmission:** This advantage is a consequence of the first; smaller file size implies easier and faster data transmission.
3. **Noise Reduction:** In certain audio compression techniques, corrupting noise signals are eliminated in the compression process, thereby improving audio fidelity.

CHAPTER 8

FUTURE SCOPE

Audio compression plays a prominent role in a lot of signal processing. There are already various audio compression techniques available at our disposal. These methods can, however, be further improved. Noise elimination techniques in audio compression can be enhanced further. Algorithms for very large compression ratios without a noticeable drop in audio quality can be developed. Algorithms to minimize loss of a fraction of the signals can also be developed.

CHAPTER 9

CONCLUSION AND DISCUSSION

The aim of the project in discussion was successfully met, audio signal compression program codes were written for both, Haar and Daubenches algorithms, and the cods were executable. The objective compression ratio of two was also met. Thus, two methods of lossless audio compression were successfully carried out, without significant data loss.

Daubenches algorithm proved to be a better algorithm for audio compression, out of the two algorithms in discussion. It gave the user the liberty of setting different window sizes and also was less lossy, as compared to Haar algorithm.

AUDIO COMPRESSION USING WAVELETS

ORIGINALITY REPORT

18%	%	18%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	James F. Peters. "Computational Proximity", Springer Science and Business Media LLC, 2016 Publication	4%
2	Occhipinti, Cristina(Guerrini, Carla). "Analisi di segnali audio mediante funzioni wavelet", AMS Tesi di Laurea - AlmaDL - Università di Bologna, 2010. Publication	3%
3	Walker, . "Haar Wavelets", Studies in Advanced Mathematics, 1999. Publication	3%
4	Monika Hadas-Dyduch. "Efficiency of Authored Mixed Prediction Model with Application to the Labor Market", Engineering Management Research, 2018 Publication	2%
5	Michael Paluszek, Stephanie Thomas. "Chapter 4 Interactive Graphics", Springer Science and Business Media LLC, 2015 Publication	2%

6	Valentyn I. Prokhorenko. "Coherently-controlled two-dimensional photon echo electronic spectroscopy", Optics Express, 06/08/2009 Publication	1%
7	C. C. Sobin, V. M. Manikandan. "A Secure Audio Steganography Scheme using Genetic Algorithm", 2019 Fifth International Conference on Image Information Processing (ICIIP), 2019 Publication	1%
8	Ajaya Shrestha, Arun Timalsina. "Color image steganography technique using daubechies discrete wavelet transform", 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2015 Publication	1%
9	"Computer Applications for Communication, Networking, and Digital Contents", Springer Science and Business Media LLC, 2012 Publication	<1%
10	"Computer Networks and Intelligent Computing", Springer Science and Business Media LLC, 2011 Publication	<1%
11	Liangliang Lu, Lijun Xia, Zhiyu Chen, Leizhen Chen et al. "Three-dimensional entanglement on a silicon chip", npj Quantum Information, 2020	<1%

	Publication	
12	Srinivas Ramavath, Rakhesh Singh Kshetrimayum. "Analytical calculations of CCDF for some common PAPR reduction techniques in OFDM systems", 2012 International Conference on Communications, Devices and Intelligent Systems (CODIS), 2012 Publication	<1 %
13	Al Frady , Alaa Saud. "Statistical and Operations Researches Models for Solving Discrimination Problems = النماذج الإحصائية وطرق بحوث العمليات في حل مشاكل التصنيف", King Abdulaziz University : Scientific Publishing Centre, 2016 Publication	<1 %
14	AgbInyA, . "VoIP : Voice over Internet Protocol", IP Communications and Services for NGN, 2009. Publication	<1 %
15	Walker, . "Haar wavelets", Studies in Advanced Mathematics, 2008. Publication	<1 %
16	F. BRACKX. Chinese Annals of Mathematics, 2003 Publication	<1 %
17	Jan Lang. "Modular Eigenvalues of the Dirichlet p(·)-Laplacian and Their Stability", Spectral Theory Function Spaces and Inequalities, 2012	<1 %
	Publication	

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		