ABSTRACT

Doppler Effect is the relative change in frequency when a source of sound moves towards or away from the observer/listener. Doppler Effect is frequently observed when a loud siren of an ambulance moves past the listener. Doppler Effect causes the frequency to shift higher or lower according to the distance, speed and direction of the source compared to the observer. (U. Zolder, 2002, p.145). The following described program creates an audio effect much different from the once found in the market. Doppler effects are frequently not used when compared to other effects such as reverberation by audio engineers. The program described in this report creates an audio effect which implements Doppler Effect useful for audio engineers for audio production.

1. INTRODUCTION

There are a number of Doppler effects implemented in digital audio using various programming languages like Matlab. This report is concerned with a Doppler shift effect. This program acts like a synthesizer, where a number of frequencies are selected and a Doppler shift is added to the frequencies and added together to create a user variable Doppler effect. The user can modify the effect by varying the speed of sound, amplitude, velocity of the source, distance from the observer and time for which the effect should run. After the Doppler Effect the user has an option to add delay effects such as reverberation and echo. The user has an option to vary the delay value hence the user can make an echo effect or delay effect according to the user’s desires.

2. FUNCTIONS

There are 4 functions in this program the main function is named newdoppler.m. This function accepts an audio wave input and also calls the other functions for various purposes during the program execution.

During the first part of the function an audio of wav format is accepted to the program. Using wavread the wav file is read and also the sampling frequency is defined. After accepting the audio file, reading it in the program and saving it under an array named ‘inputsignal’, the pitchmarker function is called.

The pitchmarker function is defined in the pitchmarker.m file. The pitchmarker function usually is used for different purposes but in this program the pitchmarker function is used to get some random frequencies so that it can be Doppler shifted. (Joshua Patton, 2011)

The pitchmarker function returns an array of frequency values and stored in the array ‘rec’.
After obtaining the values for array ‘rec’ the user can decide to use the default values or to enter the values manually for speed of sound, amplitude, velocity of the source, distance from the observer and time for which the effect should run.

The code “t = -T/2:1/fs:T/2” in the newdoppler function gives time as an array where T is the time for which the Doppler effect should run.

The code “Va=linspace(-Vs, Vs, ln)” in the newdoppler function helps in determining the velocity of the source with respect to the observer. linspace function returns an array according to the length determined by the user. ‘Vs’ is the velocity of the source determined by the distance to be travelled and the time required to travel the distance.

\[ f_d = f_s \left( 1 + \frac{C_s}{c} \right) \]

The above shown formula describes the change in frequency with respect to change in speed of sound, frequency of source and velocity of the source, which is the effect called Doppler’s effect.

‘fd’ is the change in frequency, ‘fs’ is the source frequency, ‘c’ is the speed of sound and ‘Cs’ is the velocity of the source.

The code “Fa = f0.*((c+Va)./c);” Doppler shifts the frequency according to the velocity of the source and the frequency produced by the source. In the equation c is the speed of sound, Va is an array which contains the velocity of the source at various positions, and f0 is the frequency that is to be shifted. Fa is the variable which contains the shifted Doppler frequency.

Code “wav = env.*cos( Wa.* t);” creates an array which is a cosine waveform.

The waveform is normalized so that it does not clip while writing the wave into a file and while playback.

In the next part of the program two other functions can be called. These functions are called only if the user decides to call them. If the user wants to add a delay effect such as an echo or reverberation the function “delay_” function is called. Function “delay_” needs the audio wave and the sampling frequency given while function call.

If the user wants to add the “wah wah” effect, the function “wah_wah” is called. The function “wah_wah” needs the audio wave and sampling frequency during function call.

The function delay_

The user has an option to enter the delay value. Echo is an effect which has high delay value whereas reverberation has lower delay value.

An array of zeroes is created according to the delay value entered. Zeroes are added to the start audio wav and stored in “signal_delayed” variable. Now the audio and signal_delayed variable is added. The resulting output will be an audio wave with a delay effect.

Normalization is done to the resulting wave so that the audio wave does not clip while playback and during wave write.
The function **wah_wah**
The wah wah function requires an audio wave and sampling frequency to be sent during the function call. (U. Zolder, 2002,p.55)

![Diagram of wah wah function](image)

The wah_wah code can be explained in the following. Initially a triangle wave is created. This is created so that the centre frequency of the time varying band pass filter can be modulated. The next step is to implement a filter, state variable filter. If the centre frequency is within the state variable loop repeated recalculation are done. (Cardiff university, 2006)

```matlab
Fc=minf:delta:maxf;
while(length(Fc) < length(x) )
    Fc= [ Fc (maxf:-delta:minf) ];
    Fc= [ Fc (minf:delta:maxf) ];
end
```

The above 5 lines of code help in creating the triangular wave but since the triangular wave may not be in sync with the input waveform, trimming is done. Trimming is done using the following line of code.

```matlab
Fc = Fc(1:length(x));
```

Here length(x) gives length of the input waveform. Input waveform is stored in the variable x.

\[
\begin{align*}
yl(n) &= F1yb(n) + yl(n - 1) \\
yb(n) &= F1yh(n) + yb(n - 1) \\
h(n) &= x(n) - yl(n - 1) - Q1yb(n - 1)
\end{align*}
\]

The above shown difference equations define a state variable filter.
3. **OUTCOME**

The initial outcome of the program is a graph from the pitchmarker function. The figure 3.1 shown below shows the graph from pitch marker function. The graph shows the various pitches marked by the pitch marker function. These pitch values are returned to newdoppler function for other processes.

![Figure 3.1](image1.png)

*Figure 3.1, plotting the graph from pitch marker function*

The figure 3.2 shown below shows the graphical representation of the Doppler effect. The graph shows the change in amplitude with respect to change in position of the source position. As the source gets closer the amplitude increases.

![Figure 3.2](image2.png)

*Figure 3.2, Graphical representation of Doppler effect produced by the program.*
Figure 3.3 shown below is a graphical representation of the waveform after adding the wah wah effect and the delay effect. Red graph represents the original waveform which has a a Doppler effect on it. Blue waveform is the waveform containing wah wah and reverberation effect added to it.

Figure 3.3, Graphical representation of the output from the wah_wah function. The figure show the Doppler Effect followed by wah wah effect. Red graph is the original wave and blue graph is the resulting graph.
4. REFERENCES


http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/06_CM0340Tut_MATLAB_DAFX.pdf

http://joshpatton.org/yeshua/Elec484/Elec484_files/