Course Name: Digital Signal Processing Design

Course Number and Section: **14:332:447:01**

Dereverberation: Removing Unwanted Echoes and Reverb from Recorded Audio

Part 3 of 6: FIR Comb Filter

Submitted by: Lance Darragh

Digital Audio Effects, Comb Filter

Now that we have the recovered recording we'll add a few audio effects. The algorithm used to remove the echo can be modified to be used as a creative effect. Instead of removing an echo from a recording we can add one.

(25) $y(n) = x(n) + ax(n - D)$

where again a represents the strength of the repeated copy and is between zero and one. This filter has an interesting frequency response. Taking the z-transform and solving for H(z) we obtain:

(26) $H(z) = 1 + az^{-D}$

substituting $z = e^{j\omega}$ and taking the magnitude,

(27) $H(e^{j\omega}) = 1 + ae^{-j\omega D}$

(28) $$|H(\omega)| = \sqrt{1 + 2acos(\omega D) + a^2}$$

which has zeros at $\omega = a^{1/D}e^{\pi j(2k + 1)/D}$ and peaks at $\omega = 2\pi k/D$, making a "comb" of filters alternating between attenuating and increasing amplitude, which mimics constructive and destructive interference at selected frequencies being introduced into the audio signal.

This filter can be further modified a superposition of comb filters to have an even more interesting frequency response by adding additional copies with varying attenuation into the signal.

(29) $y(n) = x(n) + ax(n - D) + a^2x(n - 2D) + a^3x(n - 3D)$

whose transfer function is:

(30) $$H(z) = 1 + az^{-D} + a^2z^{-2D} + a^3z^{-3D} = \frac{1 - a^4z^{-4D}}{1 - az^{-D}}$$

Using this filter, and a value of a = 0.45, and D = 4000 (Note: $D*f_s$ = 250msec) we can introduce a pleasing characteristic to the sound, a more subtle yet desirable form of reverberation. Appendix A.3 has the MATLAB code. The execution time for the sample-by-sample (circular delay-line buffer) method is 0.083497 seconds, which is about 13.5 times faster than MATLAB's built-in filter function, which is 1.131073 seconds.

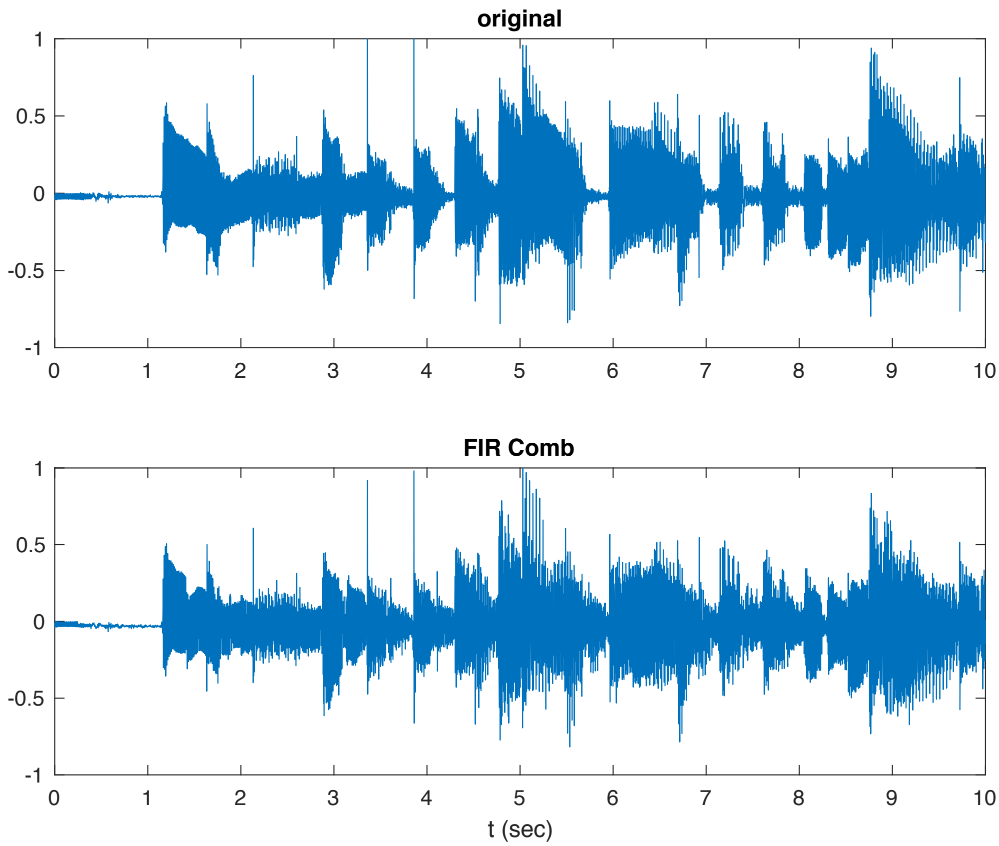
original
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10
FIR Comb
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10
t (sec)

Table of Contents

Appendix A.3: FIR Comb Filter

```
% Read in original audio
[s,fs] = audioread('original.wav');
% Set delay value in samples
D = 0.25*fs;
% Relative strength of delay
a = 0.45;
% Determine the length of the signal
[N,k] = size(s);
% Internal delay buffer for s(n)
w = zeros(1, 3*D + 1);
% Delay buffer index variable
q = 1;
% Delay buffer taps
tap1 = D + 1;
tap2 = 2*D + 1;
tap3 = 3*D + 1;
% Loop through input signal
for n = 1:N
    % Read input into w.
    w(q) = s(n);
    % y(n) = s(n) + as(n-D) + a^2s(n-2D) + a^3s(n-3D)
    y(n) = w(q) + a*w(tap1) + a^2*w(tap2) + a^3*w(tap3);
    q = q - 1;                      % Backshift index
    if q < 1                        % Circulate index
        q = 3*D + 1;
    end
    tap1 = tap1 - 1;                 % Backshift tap1
    if tap1 < 1                     % Circulate tap1
        tap1 = 3*D + 1;
    end
        tap2 = tap2 - 1;            % Backshift tap2
    if tap2 < 1                     % Circulate tap2
        tap2 = 3*D + 1;
    end
        tap3 = tap3 - 1;            % Backshift tap3
    if tap3 < 1                     % Circulate tap1
        tap3 = 3*D + 1;
    end
end

% Normalize y(n)
ymax = max(y);
```

```
y = y/ymax;

% Playback the results
sound(y,fs);
```

Plot the Signals

```
t = 1:160000;
subplot(2,1,1)
plot(t/fs,s); title('original')
subplot(2,1,2)
plot(t/fs,y); title('FIR Comb'), xlabel('t (sec)')
```

Repeating Calculation with filter Function

```
% Create impulse response
h = [1,zeros(1,D - 1),a,zeros(1,D - 1),a^2,zeros(1,D - 1),a^3];
% Output
yfilt = filter(h,1,s);

% Normalize yfilt(n)
yfiltmax = max(yfilt);
yfilt = yfilt/yfiltmax;

% Compare the results
sound(yfilt,fs);
```

Output the Results

```
audiowrite('FIRComb.wav',y,fs);
```

Published with MATLAB® R2017b