1 Abstract

2 Body

1. We begin by record the phrase 'I love Matlab':

```
fs = 10000;
r = audiorecorder(fs,16,1); % Record at fs = 10000
recordblocking(r, 8);
```

2. And saving the speech as a '.wav' file. Note this process is repeated for any other sound recording file used.

```
audiowrite('holum.wav', r.getaudiodata, fs); % Save data as .wav
```

3. We used the templates provided online as a template for constructing a method of determining voiced, unvoiced, and silenced portions of a recording that used both energy levels and zero-crossings. In particular, we defined a function that took a recording, sample rate, zero-crossing threshold, energy threshold, and a silence energy threshold, that separated the given vector in three equally sized vectors containing the results.

   The method is documented in:

   `https://www.asee.org/documents/zones/zone1/2008/student/ASEE12008_0044_paper.pdf`

```
zc_threshold = 0.100; % zero_crossings / num_samples
en_threshold = 0.075; % Energy threshold
silence_en_threshold = 0.03; % Silence energy threshold

sound_file = 'holum.wav';
[speech, fs] = audioread(sound_file);

% Identify voiced, unvoiced, and silence from basic recording
[voiced, unvoiced, silence, t] = part_3(speech, fs, zc_threshold,
        en_threshold, silence_en_threshold);

hold off;
plot(t, voiced, 'b');
hold on;
plot(t, unvoiced, 'r');
plot(t, silence, 'g');
legend('voiced', 'unvoiced', 'silence')
xlabel('time (s)');
ylabel('Amp');
title('Plot of Voiced/Unvoiced/Silence for Standard Recording');
```

4. Adding white noise to the speech, then repeating section three:

```
[sp, fs] = audioread('holum.wav');
noisy = AddNoise(sp, 5);
```

```
sound_file = 'noisy_speech.wav';
[speech, fs] = audioread(sound_file);

zc_threshold = 1; % zero_crossings / num_samples
en_threshold = 0.075; % Energy threshold
silence_en_threshold = 0.03; % Silence energy threshold

[voiced, unvoiced, silence, t] = part_3(speech, fs, zc_threshold,
    en_threshold, silence_en_threshold);

hold off;
plot(t, voiced, 'b');
hold on;
plot(t, unvoiced, 'r');
plot(t, silence, 'g');
legend('voiced', 'unvoiced', 'silence')
xlabel('time (s)');
ylabel('Amp');
title('Plot of Voiced/Unvoiced/Silence for Noisy Recording');
```

5. Obtaining the spectrum of speech in short segments.
6. Locating unvoiced segments lost in the noisy signal
7. Labeling voiced/unvoiced segments based on spectrum
8. Moving average filters
9. Repeating part 3 on a filtered noise sample

3 Conclusion

4 Discussion

5 Code

- Part-3 function for picking and returning voiced, unvoiced, and silenced portions of an audio sample.

```
function [voiced, unvoiced, silence, t] = part_3(speech,
        fs, zc_threshold, en_threshold, silence_en_threshold)
% This computes voiced, unvoiced, and silence arrays passed in through the
% Specified limits

    close all;
    clear plot;

    x = 1:length(speech);
    t = x./fs;

    f_size = 128;
    len = length(speech);
    num_F = floor(len/(f_size));
```

```
    beg = 1;
    en = f_size;

    for i = 1:num_F
        speech_frame = speech(beg:en);

        % Compute zero crossings and energy for the frame
        zc = zero_crossings(speech_frame);
        theta = sum(abs(speech_frame))/length(speech_frame);
        energ(beg:en) = theta;
        crossing(beg:en) = zc;

        % Check for zc threshold
        ts = zc/f_size <= zc_threshold;
        es = theta >= en_threshold;
        voiced_i(beg:en) = (ts && es);

        % 'Silence' is low energy
        silence_i(beg:en) = theta < silence_en_threshold;

        % Check for energy threshold
        beg = beg + f_size;
        en = en + f_size;
    end

    [voiced, unvoiced] = split_vectors(speech, voiced_i);
    [silence, unvoiced] = split_vectors(unvoiced, silence_i);

end
```

- Zero-crossing Function used in part 3.

```
function num_crossings = zero_crossings(xx)
% Count and return the number of 0 crossings in the vector, xx.

num_crossings = 0;
for i = 1:length(xx) - 1
    num_crossings = num_crossings + abs(.5 * sign(xx(i)) - .5 * sign(xx(i+1)));
end
```